“华为杯”第十五届中国研究生
数学建模竞赛

题 目　机场新增卫星厅对中转旅客影响的评估方法

摘　　要：

本文以机场新增卫星厅 S 对中转旅客影响作为研究实例，考虑航班-登机口分配问题。递进式地考虑了多种约束条件下的分配优化方案①：仅考虑航班时间安排；②：在①的基础上考虑中转旅客的总体流程时间；③：进一步细化②中考虑的中转旅客的转乘时间。对航班-登机口的分配问题在不同考虑因素的叠加下给出了分配方案。结果如下：

问题一：在不考虑乘客换乘时间和失败率以及忽略停在临时停机口的航班的前提下，我们对 20 号共 303 个转接记录飞机进行登机口分配。其中，转接记录中的 303 架飞机中有 **253 架飞机**安排到了合适的登机口，只有 **50 架飞机**未安排而停放在临时机位。飞机成功分配登机口的分配率为 **83.50%**。而登机口的使用上，69 个登机口使用了 **67** 个，只有 S29 和 S30 两个登机口未被使用。**有 34 个登机口的时间占用率率是超过 60%的，约占登机口数的一半，因此整个登机口的分配在时间轴上是非常紧凑的**。

问题二：在问题一的基础上考虑乘客的中转流程时间。在本问题的讨论上，由于“安排到临时机位的飞机数”仍然是考虑的首要目标，但是过分考虑又会忽略乘客的中转流程时间。本问题我们提出一个可容忍裕度，即“安排到临时机位的飞机数”这个目标取值大于当前求解的最优值某个裕度，也是认为满意的。在该基础上，可以尽可能地最小化客的中转流程时间。我们求解到，同样有 **50 架飞机**没有安排合适登机口且登机口使用数为 **67** 的情况下，因为问题二采取的目标函数相比于问题一增加了时间上的损失，因此旅客总体最短流程时间也从 65555 分钟降到了 **65015 分钟**，旅客平均最短流程时间也从 33.80 分钟下降到 33.54 分钟。而**在可容忍裕度范围内（即“安排到临时机位的飞机数”在 52 以内），旅客的总体最短流程时间下降到 60865 分钟**，当然这是在稍微牺牲首要目标的损失值前提下得到的，只要在可容忍裕度内，我们认为都是满意的解。

问题三：问题三是对问题二的换乘时间进行进一步的细化，考虑乘客换乘的紧张度。为了能够更好地最小化总体紧张度，我们同样考虑可容忍裕度。我们求解到。同样有 **50 架飞**

机没有安排合适登机口且登机口使用数为 **67** 的情况下，因为问题三采取的目标函数相比于问题一增加了总换乘紧张度的目标，因此旅客的总体换乘紧张度也从 400.03 下降为 **396.29 分钟**，旅客的总体换乘时间也从 112646 分钟下降为 111333 分钟。而**在可容忍的裕度范围内（即“安排到临时机位的飞机数”在 52 以内），旅客的总体换乘紧张度更是下降到 371.49 分钟**，下降幅度较大，而仅仅稍微牺牲了“安排到临时机位的飞机数”这个目标的损失。

本文创新点：

（1）、对于问题一，本文分析了目标 1（尽可能多地分配航班到合适的登机口）与（目标 2 尽可能少用登机口）之间的数量关系和从属关系。通过将目标 1 乘以目标 2 的上界，与目标 2 累加的方式，将两目标优化问题转换为单目标优化问题进行求解，同时也保证了目标 1 支配目标 2 的从属。直接将该问题进行单目标优化求解。

（2）、在针对第一问的求解过程中，本文递进式地针对问题的特点设计了搜索策略：首先，设计了启发式搜索算法；然后在此的基础上设计了引入了概率进行搜索，最后，最终综合基于启发式的规则和概率的搜索方式，结合本题的特点及遗传算法保留最优个体产生子代，不断迭代搜索的优良特性，设计了引入决策线的概念，并通过保留最佳决策线和对决策线进行变异产生可行解的随机搜索算法。该方法在启发式的前提下拥有一定的推广和泛化能力和跳出局部最优解的能力。迭代速度快，对于问题 1，所设计的搜索算法 1s 可进行 20 次迭代，迭代近千次后能够得到合理并满意的结果。

（3）、针对多目标问题二和三，在搜索结果过程中，对占支配地位的目标（尽可能多地分配航班到合适的登机口）与从属目标时间的关系处理上，本文允许牺牲占支配地位的最优目标的一部分性能从属目标性能的改进，同时这种牺牲也增大了最佳决策线的更新率，扩大了解的搜索范围。

关键词：航班-登机口分配，中转旅客换乘，启发式搜索，贪婪算法，遗传算法，parato 解集

1. 问题重述

1.1 问题的背景

由于旅行业的快速发展，某航空公司在某机场的现有航站楼 T 的旅客流量已达饱和状态，为了应对未来的发展，现正增设卫星厅 S。但引入卫星厅后，虽然可以缓解原有航站楼登机口不足的压力，对中转旅客的航班衔接显然具有一定的负面影响。本题通过建立数学模型来优化分配登机口，分析中转旅客的换乘紧张程度，为航空公司航班规划的调整提供参考依据。

飞机在机场廊桥（登机口）的一次停靠通常由一对航班（到达航班和出发航班，也叫“转场”）来标识。航班-登机口分配就是把这样的航班对分配到合适的登机口。所谓的中转旅客就是从到达航班换乘到由同一架或不同架飞机执行的出发航班的旅客。

单纯的航班-登机口的优化分配问题已经被很好地解决 [1]，Sabre Airline Solutions® 有非常成熟的产品满足航空公司和机场地勤服务公司的需求。但在优化分配登机口的同时考虑最小化旅客行走时间，学界研究有限，市场上产品一般也不具备此一功能。

机场布局中，航站楼 T 具有完整的国际机场航站功能，包括出发、到达、出入境和候机，有 28 个登机口。卫星厅 S 是航站楼 T 的延伸，可以候机，没有出入境功能，有 41 个登机口。为叙述方便起见，我们统称航站楼 T 和卫星厅 S 为终端厅。T 和 S 之间有捷运线相通，可以快速往来运送国内、国际旅客。假定旅客无需等待，随时可以发车，单程一次需要 8 分钟。

登机口分配中，登机口属于固定机位，配置有相应的设备，方便飞机停靠时的各种技术操作。航班-登机口的分配需要考虑如下规则：

a)T 和 S 的所有登机口统筹规划分配；

b) 每个登机口的国内/国际、到达/出发、宽体机/ 窄体机等功能属性事先给定，不能改变。飞机转场计划里的航班只能分配到与之属性相吻合的登机口；

c) 每架飞机转场的到达和出发两个航班必须分配在同一登机口进行，其间不能挪移别处；

d) 分配在同一登机口的两飞机之间的空挡间隔时间必须大于等于 45 分钟；

e) 机场另有简易临时机位，供分配不到固定登机口的飞机停靠。假定临时机位数量无限制。

注：本题数据中使用到的宽窄飞机型号分别有：

宽体机（Wide-body）：332, 333, 33E, 33H, 33L, 773

窄体机（Narrow-body）：319, 320, 321, 323, 325, 738, 73A, 73E, 73H, 73L。

旅客流程中，旅客流程可以按始发旅客、终到旅客和中转旅客分类规范。但由于新建卫星厅对始发旅客和终到旅客影响甚微，故不在研究范围内。中转旅客从前一航班的到达至后一航班的出发之间的流程，按国内（D）和国际（I）、航站楼（T）和卫星厅（S）组合成 16 种不同的场景。这些场景的最短流程时间和捷运乘坐次数由下表给出，其中每一格的第一个数是最短流程时间（分钟），第二个数是捷运乘坐次数。捷运时间和旅客行走时间不计入最短流程时间。

到达 \ 出发		国内出发（D）		国际出发（I）	
		航站楼 T	卫星厅 S	航站楼 T	卫星厅 S
国内到达（D）	航站楼 T	15/0	20/1	35/0	40/1
	卫星厅 S	20/1	15/0	40/1	35/0
国际到达（I）	航站楼 T	35/0	40/1	20/0	30/1
	卫星厅 S	40/1	45/2	30/1	20/0

1.2 问题的提出

问题一：本题只考虑航班-登机口分配。作为分析新建卫星厅对航班影响问题的第一步，首先要建立数学优化模型，尽可能多地分配航班到合适的登机口，并且在此基础上最小化被使用登机口的数量。本问题不需要考虑中转旅客的换乘，但要求把建立的数学模型进行编程，求最优解。

问题二：考虑中转旅客最短流程时间。本问题是在问题一的基础上加入旅客换乘因素，要求最小化中转旅客的总体最短流程时间，并且在此基础上最小化被使用登机口的数量。本题不考虑旅客乘坐捷运和步行时间，但也要求编程并求最优解。

问题三：考虑中转旅客的换乘时间。新建卫星厅对航班的最大影响是中转旅客换乘时间的可能延长。因此，数学模型最终需要考虑换乘旅客总体紧张度的最小化，并且在此基础上最小化被使用登机口的数量。本问题可以在问题二的基础上细化，引入旅客换乘连接变量，并把中转旅客的换乘紧张度作为目标函数的首要因素。和前面两个问题一样，本问题也要求把建立的数学模型进行编程，并求最优解。换乘紧张度定义为：

换乘紧张度 = 旅客换乘时间/航班连接时间

旅客换乘时间 = 最短流程时间 + 捷运时间 + 步行时间

航班连接时间 = 后一航班出发时间 - 前一航班到达时间

其中，行走时间由下列表格查找（单位：分钟。捷运乘坐时间需另行计算）

登机口区域	T-North	T-Center	T-South	S-North	S-Center	S-South	S-East
T-North	10	15	20	25	20	25	25
T-Center		10	15	20	15	20	20
T-South			10	25	20	25	25
S-North				10	15	20	20
S-Center					10	15	15
S-South						10	20
S-East							10

2. 模型的假设

- 假设一：20 号 0 点时 69 个登机口的初始状态为未被飞机停放；
- 假设二：旅客上下飞机的时间忽略不计；
- 假设三：在问题二、三中，计算旅客总体流程时间和总体换乘紧张度时，仅考虑到达和出发航班均被分配了固定登机口的中转旅客；
- 假设四：在问题二、三中，被考虑在内的中转旅客中转失败时，将其中转时间设为 6 小时；
- 假设五：假设临时机位数量无限制，且不考虑停在临时机位的航班上的旅客。

3. 符号说明

符号	意义
y_{ik}	第 i 架飞机分配至第 k 登机口，$i \in \{1,2,...,N\}$，$k \in \{1,2,...,M\}$
D_i	第 i 架飞机的离开时间
A_j	第 j 架飞机的到达时间，$j \in \{1,2,...,N\}$
G_k	第 k 个登机口的宽窄类型
P_i	第 i 个转场记录中飞机的宽窄机型
Ha_k	在第 k 个登机口到达的类型 (国际 I 或国内 D)
Hd_k	从第 k 个登机口出发的类型 (国际 I 或国内 D)
Fa_i	第 i 架飞机到达的航班类型 (国际 I 或国内 D)
Fd_i	第 i 架飞机出发的航班类型 (国际 I 或国内 D)
T_p^{ftrans}	第 p 个中转旅客记录号的旅客乘坐的两个航班连接时间，$p \in \{1,2,...,n^{travel}\}$
Nt_p	第 p 个中转旅客记录号中的旅客的数量
T_p^{flow}	第 p 个中转旅客记录号的旅客的最短流程时间
T_p^{mrt}	第 p 个中转旅客记录号的旅客乘坐捷运的时间
T_p^{wk}	第 p 个中转旅客记录号的旅客的步行时间
	其他未列出符号将在文中进行说明

4. 问题的分析

本文研究的是航班-登机口分配问题和机场新增卫星厅 S 对中转旅客影响的模型。根据现有的飞机转场的记录、出发的日期时间以及航班类型和到达的日期时间以及航班类型、旅客乘坐的航班信息和终端厅能接纳的航班类型及机体类型等数据，科学合理地优化分配登机口及降低中转旅客的换乘紧张度，为航空公司航班规划的调整提供有效的参考依据。

基于题目要求，只需要对 20 日到达或出发的航班和旅客进行分析，因此，从附件数据（InputData）中共选择了 303 条飞机转场的记录和 1733 位旅客信息。

4.1 问题一的分析

问题一中，只考虑航班-登机口的分配情况而无需考虑乘客换乘时间、成功率等问题。现已知排在 20 号的飞机专场记录有 303 辆飞机，而 T 航站楼和 S 航站楼共有 69 个登机口，另外还有一个不限容量的临时机位。由于停放在临时停机位会给旅客下机和登机造成不便和困难，因此停放在临时机位的飞机应尽可能少。那么我们的首要目标就是如何把 303 辆飞机尽可能分配到 69 个登机口中。在此基础上，如果机场能够腾出一些全天不用的登机口，将有利于应对各种突发状况。根据对问题一的分析，我们的目标函数考虑包含以下两点：

- 目标 1：尽可能多地分配航班到合适的登机口。
- 目标 2：尽可能少用登机口。

由于 303 辆飞机中有一部分在降落之后的停留时间非常长，超过 8 小时甚至十几个小时，并且停留时间会一直占用登机口，这将使得后来飞机无法分配到合适的登机口。我们做了一个初步的计算，如果把 303 个转记录的每一辆飞机的停留时间累加上，再算上同一登机口的两飞机之间的空挡间隔时间 45 分钟，那么相当于需要 56 个登记位每一个登机位都需要满打满算地工作 24 小时，才能够满足 303 辆飞机的转场停放需求。但是，由于航班安排时间参差不齐，飞机航班的安排存在集中到来或者集中起飞的情况，并且各个登机口和飞机之间需要国际国内航班类型匹配和宽体机/窄体机的匹配，因此要想找到一个合理并且满足要求的解是非常困难的。

根据上述的问题分析，针对第一问的求解策略主要有以下三种：

策略 1：根据问题要求构造最小化临时停机位使用次数和登机口使用数量的目标函数，引入时间约束和机型、国内国际类型的约束，采用优化算法进行全局搜索，寻找最优解。

策略 2：采用启发式的贪婪算法，确定当前时刻当前班次的最优停靠方案，搜索基于先验知识的局部最优解。

策略 3：在策略 2 的具有先验知识搜索的基础上进行改进，引入随机搜索算法，使得算法可以在启发式的前提下，具有一定的推广和泛化能力。

策略 1 具有建模简单、全局最优等优点，但需要暴力计算，耗费大量时间和资源。相反，策略 2 可以快速寻求到一个可行解，但是策略的设计需要较好的先验知识，而且只能找到局部解。策略 3 是策略 1 和策略 2 的一种折衷，如果运用地好，可以在较短时间内找到一个稍差于最优解的可行解，同时也拥有寻找最优解的可能。

本文采用第三种策略，结合了启发式的贪婪算法和随机搜索的遗传算法的思路。启发式算法保证了我们的算法搜索方向是更加正确和合理的，并在此基

础上引入了遗传算法的变异过程，将 303 个转场记录飞机的登机口安排看作一个染色体，每一个航班的到来看作一个节点或者一个基因，然后基因随机变异，并且将变异较好的路径保存下来作为母代，之后的搜索将在母代的基础上进行。该算法具有较好的搜索能力和较快的收敛速度

4.2 问题二的分析

问题二需要在问题一的基础上讨论乘客的最短流程时间。在问题一的基础上，也就是我们的首要目标仍然是尽可能地将每一个转机记录的飞机安排到合适的登机口从而尽量少地使用临时机位，然后再讨论如何安排机场航班的停靠位置以使得飞机上的旅客在中转过程所花费的时间最短，紧接着我们才考虑如何去最小化登机口的使用数量。因此我们的目标按照题目要求的重要程度可以表示如下 3 点：

- 目标 1：尽可能多地分配航班到合适的登机口。
- 目标 2：尽可能减少旅客换成过程中的最短流程时间。
- 目标 3：尽可能少用登机口。

其中，最短流程时间跟航站楼以及国内国际航班类型有关。对于多目标的优化问题我们通常需要考虑以下复合的选择：

1) 每一个目标取什么值，原问题可以得到最满意的解。

2) 每一个决策变量取什么值，原问题可以得到最满意的解。

多目标优化问题的求解不能只追求一个目标的最优化，而不顾其它目标。这种情况需要分析目标之间的关系，如果目标是方向一致的时候，可以采取同时优化的方式，当目标方向处于冲突状态时，就不会存在所有目标函数同时达到最大或最小值的最优解，此时可以求取帕累托解。对于上述三个目标函数，我们一方面可以设置不同的权重去将他们合并成一个优化目标去求解，此时往往找到的只是目标函数的最优解而非原问题的最优解。另一方面可以根据问题的依赖关系，如本问题中，需要在问题一的基础上去考虑旅客换乘的流程时间，因此可以采用贪心算法去设计相应的局部最优的决策方案，去搜索局部最优解。

问题二采用贪心算法需要考虑的是什么才是局部最优。由于贪心算法是一种逐步决策的方式，那么在当前步，我们只能考虑已知的乘客所需要的转乘时间来对登机口进行选择。一架飞机将要降落时，已知的乘客包含两个部分，一个是从该飞机下来地旅客的换乘航班已安排在某个确定登机口的旅客，另一个是将要乘坐该飞机起飞，而且已经到达机场的旅客。因为只有这部分旅客换乘的最短流程时间是知道的，因此我们选择登机口的时候需要最小化这部分时间去选择最优的登机口。这种办法可以快速找到合理的解，通常不是全局最优的。因此我们跟第一问相似，引入了变异算子，借鉴遗传算法的规则去保留好的父代，繁

衍子代，从而增强了算法的全局搜索能力。

4.3 问题三的分析

问题三与问题二相似，是问题二的细化版本，因此可以采用相同的算法去求解。问题三同样是在前面问题的基础上进行讨论，然后引入本问题重点讨论的换乘时间和紧张度问题。问题三的优化目标按照重要程度的顺序表示如下：

- 目标 1：尽可能多地分配航班到合适的登机口。
- 目标 2：尽可能减少旅客换乘过程中的时间紧张度。
- 目标 3：尽可能少用登机口。

问题三中旅客的换乘时间由更多的因素决定，除了问题二中的航站楼和国内国际航班类型外，还由捷运时间和行走时间决定，而捷运时间和行走时间跟航站楼、航班类型以及航站楼的东南西北分布相关，因此时间的计算上会变得复杂，相应的约束也增加了许多。除此之外，问题三的目标函数和类型与问题二相似，针对这种多目标的优化方法，由于约束非常多，在全局上进行搜索往往难以找到最优解，甚至无法找到一个合适的解。因此本文提出一种针对本问题的新方法，结合了贪心算法的启发功能，又利用了概率搜索的模式进行节点的变异和选择，并且将优势路径保存到父代，遗传给子代。经过检验，我们的算法能够快速找到合理满意的解，并且具有一定的全局搜索能力。

5. 问题一：航班-登机口的分配问题

如前所述，只需要考虑 20 日到达或 20 日出发的航班的信息，因此共选择了 303 条飞机转场记录进行登机口的分配。

5.1 问题一模型的建立

第一步：确定模型的决策变量：

1) 飞机是否被安排到登机口上的决策变量 y_{ik}：

$$y_{ik} = \begin{cases} 1 \\ 0 \end{cases} \tag{1}$$

其中，1 代表当且仅当第 i 架飞机被安排到第 k 个登机口，0 代表的是除此之外其他情况，这里 $i \in \{1, 2, ..., N\}$, N 代表了 303 架飞机即 303 条飞机转场记录，$k \in \{1, 2, ..., M\}$, M 代表了 69 个登机口，显然，构成了一个 303×69 的决策矩阵。

2) 飞机是否停在了临时机位，即飞机是否被安排到临时机位的决策变量 z_i：

$$z_i = \begin{cases} 1 \\ 0 \end{cases} \tag{2}$$

其中，1 代表了第 i 架飞机被安排到临时停机位。0 则反之，第 i 架飞机没有被安排在临时机位。题目中假定临时停机位数量无限制，因此对其数量不作限定，也就是对于临时机口，我们只关心第 i 架飞机是否被安排在临时机口，不关心被安排在哪个临时机口上，即 z_i 是一个 303×1 的向量。

第二步：确定模型的约束条件，此问题的约束条件如下：

1）独占性要求。不能出现同一架飞机既在登机口又在临时机口的情况，故有如下约束条件为：

$$\sum_{k=1}^{M} y_{ik} + z_i = 1, \qquad \forall i \in \{1, ...N\} \tag{3}$$

2）航班类型与登机口类型相匹配要求。其中包括了第 i 架飞机到达或出发的航班类型与其登机口类型匹配，本文将对该类型进行编码，编码类型和意义如下表1所示：

表 **1**　到达或出发的航班类型或登机口类型的编码表

编码类型	意义
-1	航班类型或登机口类型为国际航班（I）
0	登机口类型为国际航班或国内航班（D/I）
1	航班类型或登机口类型为国内航班（D）

$Ha_k \in \{-1, 0, 1\}$，$Fa_i \in \{-1, 1\}$，$Hd_k \in \{-1, 0, 1\}$，$Fd_i \in \{-1, 1\}$。显然，若航班类型与登机口类型符合对应的匹配要求时，二者类型相乘大于或等于 0 即可。例如某架飞机到达的航班类型为 I（值为-1），则可匹配的登机口类型可以为 I（值为-1）、D/I（值为 0），二者相乘后结果等于 0 满足条件，反之登机口类型为 D（值为 1）则不满足条件。综上，该约束条件为：

$$Ha_k \times Fa_i \times y_{ik} \geq 0, \qquad \forall i \in \{1, ...N\}, k \in \{1, ...M\} \tag{4}$$

3）由约束条件（2）可知，第 i 架飞机出发的航班类型与登机口类型相匹配的约束条件为：

$$Hd_k \times Fd_i \times y_{ik} \geq 0, \qquad \forall i \in \{1, ...N\}, k \in \{1, ...M\} \tag{5}$$

4) 从题目中的条件可知，第 i 架飞机的宽窄型号与登机口宽窄型号需相匹配。与约束（2）的分析类似，$G_k \in \{0,1\}$，$P_i \in \{0,1\}$，只有当飞机与登机口两者的宽窄型号一致时，二者相减才等于 0，否则不为 0。其约束条件如下所示：

$$(G_k - P_i) \times y_{ik} = 0, \quad \forall i \in \{1,...N\}, k \in \{1,...M\} \tag{6}$$

5）安全时间间隔要求。分配在同一登机口的两飞机之间的空挡间隔时间必须大于等于 45 分钟。我们规定第 j 架飞机是晚于第 i 架飞机来匹配同一登机口 k，即有第 j 架飞机的到达时间 A_j 与第 i 架飞机的离开时间 D_i 的间隔时间大于等于 45 分钟 (用变量 α 表示)，而需要作此判断的前提是第 i 架飞机和第 j 架飞机都停在同一登机口 k 上，停在不同登机口上的飞机无需此约束条件。该约束条件式子为：

$$(A_j - D_i - \alpha) \times y_{ik} \times y_{jk} \geq 0, \quad \forall\, i,j \in \{1,...N\}, i < j, k \in \{1,...M\} \tag{7}$$

其中，本文将 303 个飞机转场记录按照其到达航班的时间的先后顺序排序好，故 $i < j$ 表示了我们规定的第 j 架飞机是晚于第 i 架飞机。

第三步：确定目标函数：在满足尽可能多地分配航班到合适的登机口的基础上，即最小化停在临时机口的飞机数量，同时最小化被使用登机口的数量。显然，上述问题可视为多目标优化问题，其中最小化临时停机口飞机数量的目标（记为主目标）支配了最小化临时登记口的目标（记为子目标）。对于多目标问题，通常有两种常用解决的方法：一是，搜索 parato 解集，通过评议函数对解集进行分析筛选，得出较为满意的解决方案；二是，将多个目标函数加权为一个目标函数，按照单目标优化的问题的方法进行求解。考虑本体较为特殊的情况，即，子目标最小化登机口数量函数有界，最大不超过 69；而我们希望飞机航班尽可能被安排在临时登记口处，即 z_i 越少越好。因此，如公式（10）可将主目标函数乘以 70（大于 69 即可），与子目标相加作为待优化的单目标优化函数。如此，可保证在极大可能地优化主目标的前提下，再考虑优化子目标。

1）目标函数：最小化被安排在临时机口的飞机数量

$$min \sum_{i=1}^{N} z_i \tag{8}$$

2）子目标函数：最小化被使用登机口的数量

$$min \sum_{k=1}^{M} \frac{\sum_{i=1}^{N} y_{ik}}{\sum_{i=1}^{N} y_{ik} + \varepsilon} \tag{9}$$

其中，ε 设定为极小数，本文设定为 0.0001，防止出现分母为 0 无意义的情况。

综上所述，问题一的数学模型为：

$$min\ \ F_{q1} = (f_{q11},\ f_{q12}) = 70 \times f_{q11} + f_{q12} \tag{10}$$

$$s.t. \begin{cases} \sum_{k=1}^{M} y_{ik} + z_i = 1, & \forall i \in \{1, ...N\} \\ Ha_k \times Fa_i \times y_{ik} \geq 0, & \forall i \in \{1, ...N\}, k \in \{1, ...M\} \\ Hd_k \times Fd_i \times y_{ik} \geq 0, & \forall i \in \{1, ...N\}, k \in \{1, ...M\} \\ (G_k - P_i) \times y_{ik} = 0, & \forall i \in \{1, ...N\}, k \in \{1, ...M\} \\ (A_j - D_i - \alpha) \times y_{ik} \times y_{jk} \geq 0, & \forall\, i, j \in \{1, ...N\}, i < j, k \in \{1, ...M\} \end{cases}$$

其中，

$$\begin{cases} f_{q11} = \sum_{i=1}^{N} z_i \\ f_{q12} = \sum_{k=1}^{M} \frac{\sum_{i=1}^{N} y_{ik}}{\sum_{i=1}^{N} y_{ik} + \varepsilon} \end{cases}$$

5.2 问题一模型的求解

基于启发式贪婪概率搜索方法的问题 1 模型求解：

在下文中，我们将通过递进的方式描述针对该问题的模型求解的过程。

1）考虑到算法的运行效率，为了快速生成可行解，我们首先使用了启发式的搜索方式，对每一个依时间次序到达的航班，筛选出可用的空闲登机口中出发和到达航班类型（国内或国际）相同及可服务机型（宽窄）相匹配的。启发式的规则表述为：

1^o、到达和出发航班完全匹配的优先选择，仅有出发或达到航班类型完全匹配的航班次之，DI-DI 类型的登机口则最后选择（完全匹配即 D 与 D 航班匹配，I 与 I 航班匹配）。

2^o、当天已被使用的登机口优先匹配，未被使用的登机口次之。将上述的规则量化为权重计算，其量化方式：按照一定固定的次序，判断登机口是否可用（从而保障已使用过的登机口符合条件时首先被使用）；然后令完全匹配、半匹配、不匹配的权重分别为 2, 1, 0，对到达航班和出发航班的匹配情况累加。

最后取权重最高的作为当前飞机的登机口。

以上提到的规则，显然是合理的，即，通用的登机口如 DI-DI 类型的登机口宜最后，未被使用的登机口应该不使用，以降低代价函数。然而，这种启发式的算法是使用了先验知识，且在当前决策步中，有可用的登机口时，不具备考虑将当前航班停靠在临时停机场可能会产生更优解的情况，从而该启发式算法针对该问题无法保证得到全局最优解。

2）为了保证可能找到全局的最优解，我们在此基础上引入了概率策略，在每个决策步中，首先，以一定概率，如以 7% 的概率将航班分配至临时停机场作

为分析。有 93% 的概率分配至可用的固定登机口中，依照上述启发式规则，量化评价可用固定登机口，依权重以以赌轮盘的方法选择对应的登机口。此方法为启发式的贪婪搜索算法。然而，该算法的搜过过程过于随机，难以收敛。

3）在启发式贪婪随机搜索的基础上，我们针对该问题提出了新的方法。结合遗传算法、蚁群算法的优势，本文提出的新方法在启发式贪婪随机搜索方法的基础上，进一步考虑在每一次迭代周期中，保留优秀的个体进行变异，进一步搜索更好的解。方法介绍如下：

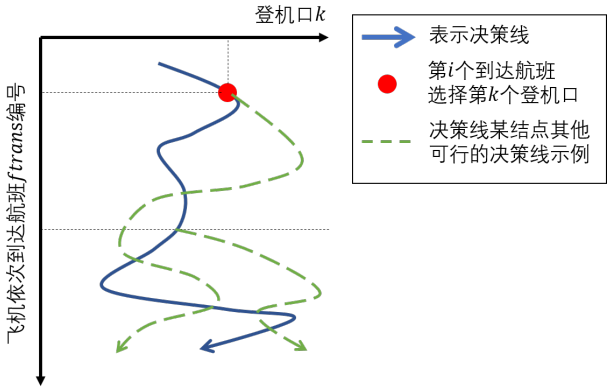


图 **1**　决策线示意图

如图1，我们首先对航班-登机口分配决策量 y_{ik} 进行分析，令决策量中每个依时间顺序排列的航班索引号为纵轴，而登机口索引号为横轴，将每个航班选择的登机口沿纵轴依次连接，可以得到一条决策线。注意，该决策线在任意一点如上图的红色点，若在下一个航班中选择了其他可用的登机口，如图中绿色虚线所示，将很大可能受登机口类型的限制沿不同的轨迹行走，可见，该问题可行解多，决策线易改变，适合以概率的方式进行搜索。然而该序列决策受到飞机类型和航班类型、登机口数量的限制，耦合性强，使用 0-1 整数规划易因约束条件（百万级）过多而求解效率低下，利用遗传算法、免疫算法、蚁群算法等易产生不符合约束条件的子代。因此在本题中，我们综合考虑上述情况后提出的搜索方式如图2所示：

搜索策略：

1^o、随机选取一个变异点 point。图中变异点（红点）为航班索引号，该点以前的航班依照当前代最优个体（最佳决策线）选择对应的登机口，该点以启发式贪婪搜索的方式确定登机口，该点以后的处理将在下文提及；

2^o、在变异点后，随机选择两个切换点 point0 和 point1。切换点即搜索策略切换点，用以切换启发式规则亦或是启发式贪婪搜索的方式选择登机口。本文规定将变异点与最后一个航班点连接成环，令 point0 到 point1 向下的方向所包含的航班使用启发式贪婪搜索的方式，其他航班选择启发式规则的方式选择登

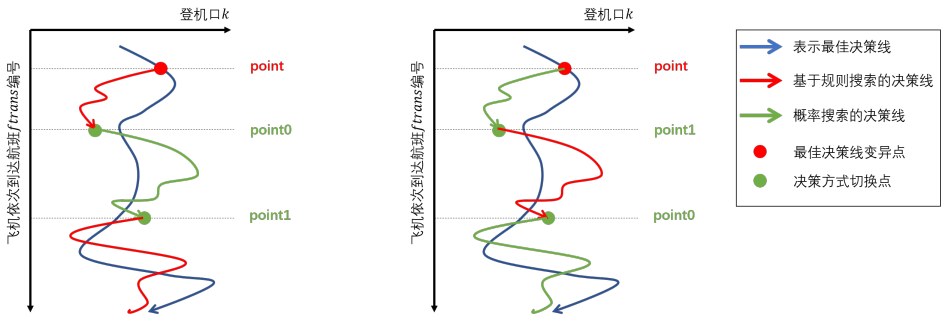


图 **2**　搜索策略示意图

机口；

3^o、所有航班搜索完毕后，对于单目标问题，如问题 1，保留最优的个体（代价函数最小的个体），并在此基础上随机生成变异点和切换点回到步骤 1^o 进行搜索。

利用上述搜索策略，得到的最佳决策线如下图3所示：

问题一求解过程算法的伪代码如下：

```
[考虑单个目标优化的伪代码]
输入:
FTrans   : 按到达航班的时间顺序排列好的转场记录
GatesInfo : 登机口信息
Tickets  : 订单记录
G        : 迭代次数G
过程:
00. 初始化最佳决策量Y_best的代价函数值cost_best=1000000
01. For g=1 to G
02.  If g==1
03.    基于设定的规则进行决策确定一个可行解作为Y_best的初始值
04.  else
05.    初始化决策变量Y_ik
06.    随机选取当前最优解Ybest决策变异节点号point
07.    随机选取局部决策变异节点号 point0 和 point1
08.    for FTrans中的每条记录 ftrans 及索引号idx
09.      if 当前索引号idx小于变异节点号point Then
10.        根据Y_best选择登机口，更新Y_ik
11.      else
12.        检测当前空闲且飞机型号、航班类型匹配的登机口gates
13.        在gates中删除Y_best选择的登机口
14.        评价每个可用的登机口，基于设定的规则计算权重weights
15.        if 索引号idx落在从point0到point1循环中 Then
16.          选择weights最大的登机口，更新Y_ik
17.        else
18.          根据weights，用赌轮盘的方式随机选择可用的登机口
19.        end if
```

```
20.       end if
21.     end for
22.     计算Y_ik对应的代价函数cost
23.     if cost<cost_best Then // 注：对多目标代价函数而言，
24.       更新Y_best,cost_best // 可增加其他方式进行评价
25.     end if
26.   End if
27. End For
输出：最佳决策变量Y_best，第i个航班分配到第k个登机口
```

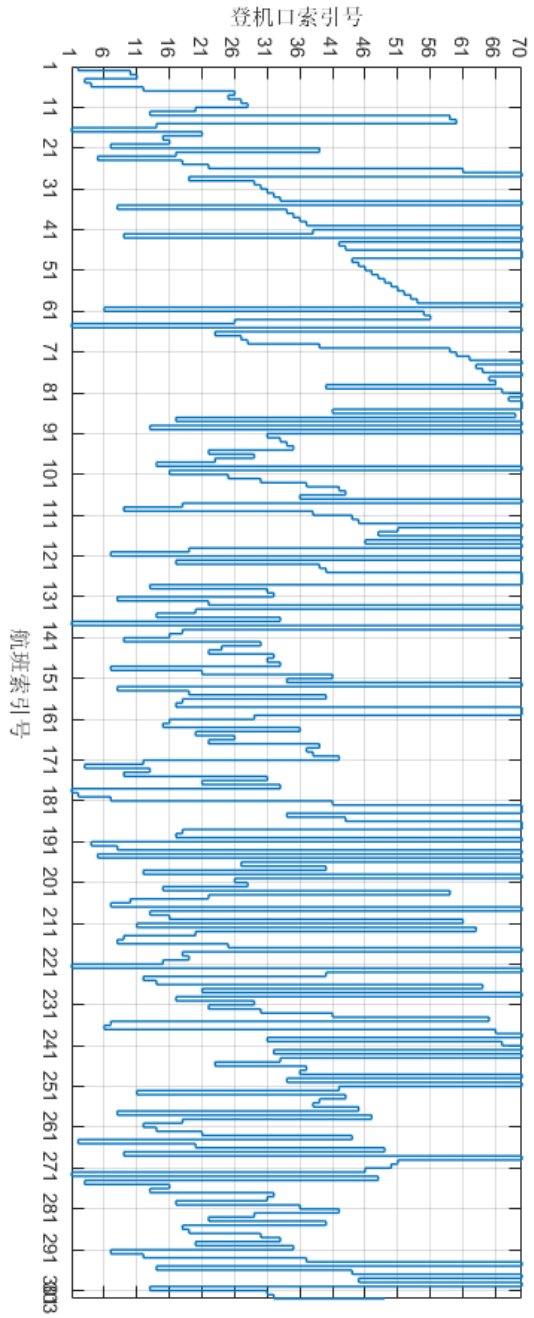


图 **3**　问题一最佳决策线示意图

5.3 问题一模型的结果

1）结果：求解后可得有 253 架飞机的转接记录可被安排到合适的登机口，50 架飞机的转接记录未被安排登机口，69 个登机口共 67 个登机口被使用，剩余 2 个登机口 S29 和 S30 未被使用。具体每架飞机航班-登机口的分配结果如表2所示：（由于篇幅有限，按照到达时间先后排序后，展示前 20 架飞机成功分配到登机口的记录）

表 **2**　问题一中飞机航班-登机口的部分分配结果

飞机转场记录号	到达航班	出发航班	对应登机口	飞机转场记录号	到达航班	出发航班	对应登机口
PK208	NV847	NV690	T2	PK112	NV6253	NV316	T20
PK062	GN0523	GN0256	T10	PK117	NV6779	NV6738	T13
PK072	GN0497	GN0644	T11	PK129	NV319	NV846	S31
PK089	NV663	NV692	T3	PK131	NV673	NV320	S32
PK094	NV693	NV662	T4	PK136	NV6753	NV6358	T14
PK102	GN0209	GN0658	T12	PK144	NV621	NV322	T1
PK104	NV697	NV840	T26	PK142	GN0237	GN920	T21
PK106	NV6489	NV880	T25	PK145	NV6317	NV6540	T15
PK107	NV601	NV664	T27	PK147	NV6725	NV6724	T16
PK108	NV821	NV608	T28	PK148	NV6549	NV306	T7

将 69 个登机口接纳 253 架飞机转接记录用图4表示，图中纵坐标代表了 69 个登机口，每一段彩色部分为某一架飞机停在了对应的登机口上的示意图，空白部分则表示该登机口没有飞机停放，横坐标为以分钟为单位的时间轴。可以从图中明显看出，每一横轴的彩色部分没有重叠且有一定的间隔（大于等于 45 分钟），验证了结果的合理性。

2）结果：而 50 架飞机的转接记录未被安排登机口中，也展示前 10 架飞机转接记录如表3所示：

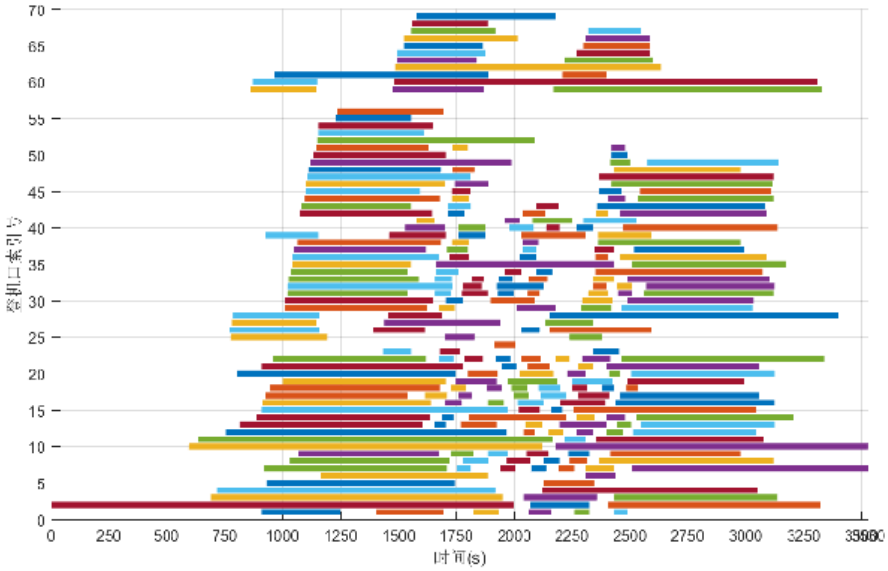


图 **4**　问题一 **69** 个登机口接纳 **253** 架飞机转接记录

表 **3**　问题一中未分配到登机口的部分飞机航班

飞机转场记录号	到达航班	出发航班	飞机转场记录号	到达航班	出发航班
PK165	NV6549	NV306	PK190	NV6549	NV306
PK177	NV6549	NV306	PK444	NV6549	NV306
PK183	NV6549	NV306	PK451	NV6549	NV306
PK186	NV6549	NV306	PK461	NV851	NV828
PK189	NV6549	NV306	PK464	NV689	NV318

5.4　问题一结果的数据分析

1）给出成功分配到登机口的航班数量和比例，按宽、窄体机分别画图。

行程包含 20 号的所有转场记录的飞机一共有 303 架次，在 20 号当天共有 606 个航班。其中有 50 架，即 100 个航班没有分配到合适的登机口（安排在临时停机位），所以分配到登机口的航班数量为 506 个，成功分配的比例为 83.50%。将成功分配到每个宽型登机口和窄型登机口的航班数量和比例分别画图，如图5、6所示

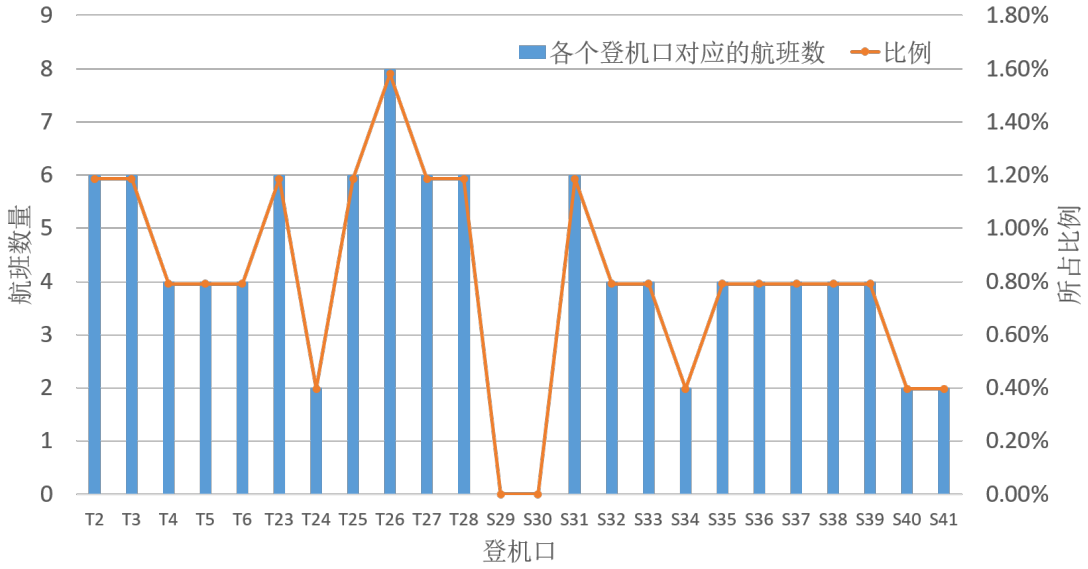


图 **5**　问题一中成功分配到宽型登机口的航班数量和比例图

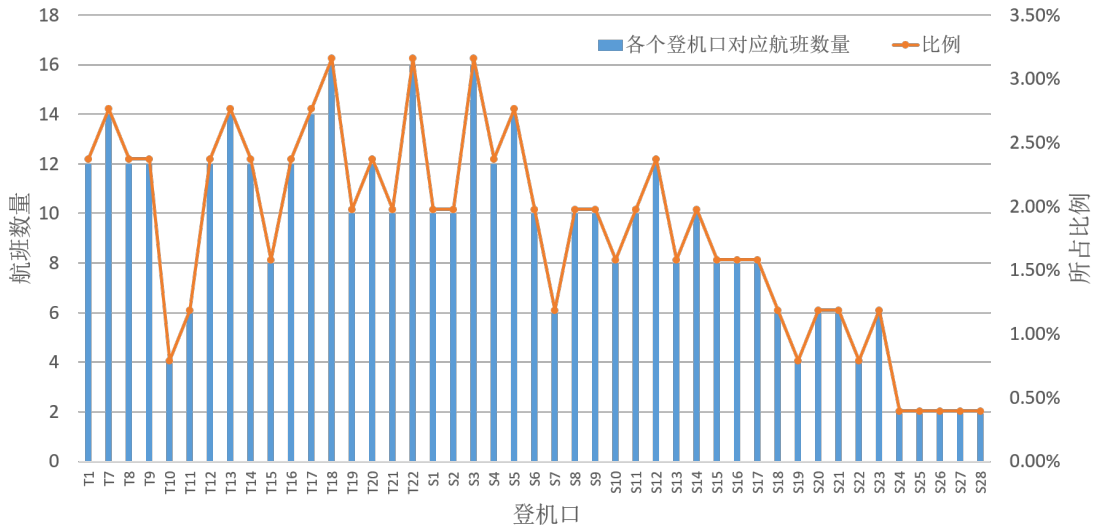


图 **6**　问题一中成功分配到窄型登机口的航班数量和比例图

2）给出 T 和 S 登机口的使用数目和被使用登机口的平均使用率（登机口占用时间比率），并画成图。

T 航站楼的登机口被使用了 28 个（总共 28 个），卫星厅 S 的登机口被使用了 39 个（总共 41 个），在 20 号一天每个登机口占用时间的比率如下图7所示，卫星厅 S29、S30 未被使用，即使用率为 0。

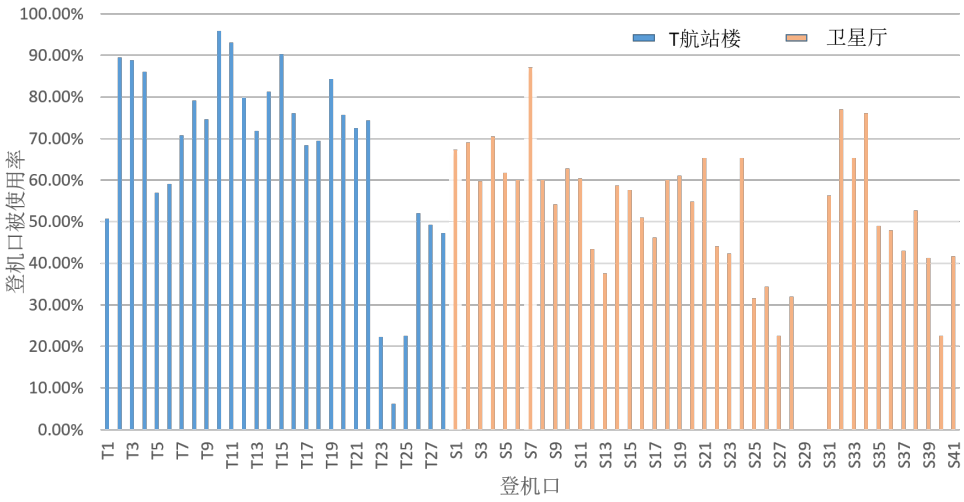


图 7　20 号一天中每个登机口占用时间的比率图

6.　问题二：中转旅客最短流程时间的问题

6.1　问题二旅客数据的分析

在问题二中，考虑了 1733 个旅客记录号，经筛选发现，共有 30 个旅客记录号的到达或出发的航班没有对应的飞机转场记录号及航班时间，因此剔除这 30 个旅客记录号。这 30 个旅客记录号分别为：T1400、T1413、T1633、T1786、T1787、T1788、T1789、T1790、T1791、T1856、T2093、T2094、T2357、T2433、T2569、T2577、T2578、T2579、T2591、T2600、T2601、T2615、T2616、T2657、T2658、T2659、T2660、T2800、T2937、T2943。剔除后剩余 1703 个旅客记录号信息。

剩余 1703 个旅客记录号中，有 54 个旅客记录号搭乘的到达航班对应的飞机转场记录不在我们的考虑范围中，即该飞机转场记录对应的到达或出发航班日期都不在 20 号，例如，旅客记录号 T1530 的旅客乘坐航班 6 月 19 号 22:05 的航班 NV3553 到达，该航班对应的飞机转场记录号为 PK176，而 PK176 对应的出发航班 NV6652 日期为 6 月 19 号 23:15，显然，该飞机转场记录号 PK176 不在本论文考虑范围，类似情况共有 54 个旅客记录号，分别为 T1530、T1531、T1532、T1515、T1516、T1517、T1518、T1491、T1519、T1483、T1492、T1533、T1474、T1459、T1475、T1460、T1480、T1481、T1482、T1520、T1476、T1450、T1451、T1452、T1453、T1454、T1421、T1422、T1455、T1477、T1478、T1479、T1461、T1462、T1463、T1408、T1423、T1409、T1410、T1401、T1402、T1403、T1404、T1390、T1449、T1381、T1382、T1383、T1391、T1424、T1425、T1378、T1379、T1317，也将其剔除，即剔除后剩余 1649 个旅客记录号信息。

6.2 问题二模型的建立

问题二是在问题一的基础上加入旅客换乘因素，要求最小化中转旅客的总体最短流程时间，并且在此基础上最小化被使用登机口的数量。当然，首先要使得尽可能多地分配航班到合适的登机口。

由假设三可知，计算旅客总体流程时间时，仅考虑到达和出发航班均被分配了固定登机口的中转旅客，而这部分被考虑的中转旅客有两种情况，一是换乘成功，即在不考虑旅客乘坐捷运和步行时间下，旅客的最短流程时间（T^{flow}）要小于乘坐的航班连接时间（T^{ftrans}）；二是换乘失败，即 T^{flow} 大于 T^{ftrans}，由假设四可知，其中转时间以 6 个小时 (360 分钟) 计入惩罚。本文用 $T_p^{success}$ 表示问题二中的旅客 p 换乘是否成功。

$$T_p^{success} = (T_p^{ftrans} - T_p^{flow}) \times \sum_{k=1}^{M} y_{p_a k} \times \sum_{k=1}^{M} y_{p_d k}, \quad \forall p \in \{1, ...n^{travel}\} \tag{11}$$

其中，p 为第 p 个旅客记录号，n^{travel} 代表中转旅客记录号的总数量。$y_{p_a k}$ 中下标 p_a 表示旅客到达时乘坐的航班，下标 p_d 表示旅客出发时乘坐的航班。$y_{p_a k}$ 表示旅客到达时乘坐的 p_a 航班是否有停在第 k 登机口，若是则为 1，否则为 0。$y_{p_d k}$ 表示旅客出发时乘坐的 p_d 航班是否有停在第 k 登机口，若是则为 1，否则为 0。旅客换乘是否成功将产生不同的最短流程时间：

$$T_p^{flownew} = \begin{cases} T_p^{flow}, \ T_p^{success} > 0 \\ 0, \quad T_p^{success} = 0 \\ 360, \ T_p^{success} < 0 \end{cases} \tag{12}$$

其中，$T_p^{flownew} = 0$ 时代表该旅客乘坐的航班没有被安排在登机口上，即这种情况下的旅客不在考虑范围内，即最短流程时间记为 0。

第一步：确定模型的决策变量（与问题一相同）：

1) 飞机是否被安排到登机口上的决策变量 y_{ik}：

$$y_{ik} = \begin{cases} 1 \\ 0 \end{cases} \tag{13}$$

与问题一相同，构成了一个 303×69 的决策矩阵。

2) 飞机是否被安排在了临时机位，即飞机是否被安排到临时机位的决策变量 z_i：

$$z_i = \begin{cases} 1 \\ 0 \end{cases} \tag{14}$$

其中，1 代表了第 i 架飞机被安排到临时停机位。0 则反之。题目中假定临时停机位数量无限制，因此对其数量不作限定，也就是对于临时机口，我们只关心第 i 架飞机是否被安排在临时机口，不关心被安排在哪个临时机口上，即 z_i 是一个 303×1 的向量。

第二步：确定模型的约束条件：

问题二是在问题一的基础上加入旅客换乘因素最短流程时间，该因素影响了目标函数，而约束条件仍然不变，与问题一的相同为：

$$s.t.\begin{cases} \sum_{k=1}^{M} y_{ik} + z_i = 1, & \forall i \in \{1,...N\} \\ Ha_k \times Fa_i \times y_{ik} \geq 0, & \forall i \in \{1,...N\}, k \in \{1,...M\} \\ Hd_k \times Fd_i \times y_{ik} \geq 0, & \forall i \in \{1,...N\}, k \in \{1,...M\} \\ (G_k - P_i) \times y_{ik} = 0, & \forall i \in \{1,...N\}, k \in \{1,...M\} \\ (A_j - D_i - \alpha) \times y_{ik} \times y_{jk} \geq 0, & \forall\, i,j \in \{1,...N\}, i < j, k \in \{1,...M\} \end{cases}$$

第三步：确定模型的目标函数：由题意可知，在问题一的基础上，也就是仍然要尽可能多地分配航班到合适的登机口，即该目标优先级最高。

1）目标函数：最小化被安排在临时机口的飞机数量

$$min \sum_{i=1}^{N} z_i \tag{15}$$

2）子目标函数 1：最小化中转旅客的总体最短流程时间

$$min \sum_{p=1}^{n^{travel}} (T_p^{flownew} \times Nt_p) \tag{16}$$

其中，$T_p^{flownew}$ 代表判断能否中转成功后，第 p 个中转旅客记录号中的旅客的最短流程时间，Nt_p 代表第 p 个中转旅客记录号中的旅客的数量。

3）子目标函数 2：最小化被使用登机口的数量

$$min \sum_{k=1}^{M} \frac{\sum_{i=1}^{N} y_{ik}}{\sum_{i=1}^{N} y_{ik} + \varepsilon} \tag{17}$$

其中，ε 设定为极小数，本文设定为 0.0001，防止出现分母为 0 无意义的情况。

综上所述，问题二的模型为：

$$min\ \ F_{q2} = (70 \times f_{q21},\ f_{q22},\ f_{q23}) \tag{18}$$

$$
s.t.\begin{cases}
\sum_{k=1}^{M} y_{ik} + z_i = 1, & \forall i \in \{1,...N\} \\
Ha_k \times Fa_i \times y_{ik} \geq 0, & \forall i \in \{1,...N\}, k \in \{1,...M\} \\
Hd_k \times Fd_i \times y_{ik} \geq 0, & \forall i \in \{1,...N\}, k \in \{1,...M\} \\
(G_k - P_i) \times y_{ik} = 0, & \forall i \in \{1,...N\}, k \in \{1,...M\} \\
(A_j - D_i - \alpha) \times y_{ik} \times y_{jk} \geq 0, & \forall\, i,j \in \{1,...N\}, i < j, k \in \{1,...M\}
\end{cases}
$$

其中，

$$
\begin{cases}
f_{q21} = \sum_{i=1}^{N} z_i \\
f_{q22} = \sum_{p=1}^{n^{travel}} (T_p^{flownew} \times Nt_p) \\
f_{q23} = \sum_{k=1}^{M} \frac{\sum_{i=1}^{N} y_{ik}}{\sum_{i=1}^{N} y_{ik} + \varepsilon}
\end{cases}
$$

6.3 问题二模型的求解

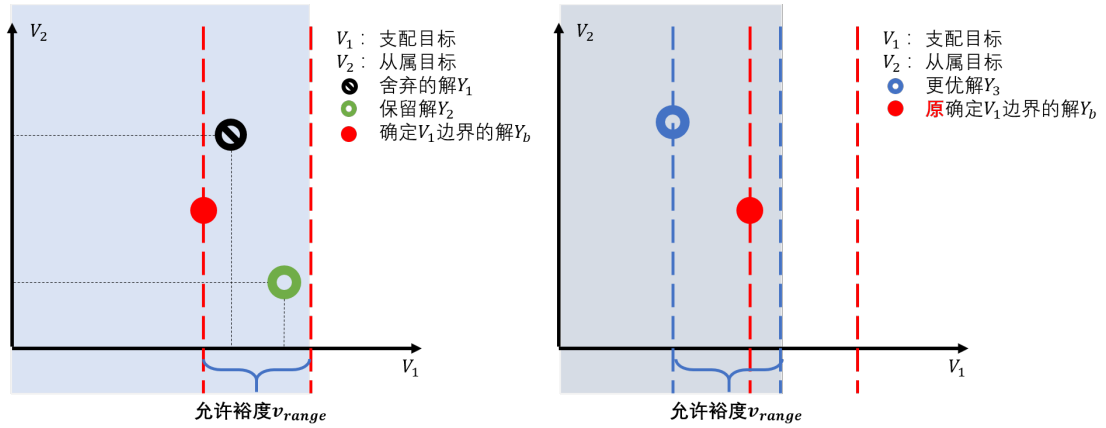
问题 2 在问题 1 的基础上考虑总体最小流程时间，问题优先级为：

最小未分配航班数目 > 最小总体流程时间 > 最少登机口数目

考虑到中转旅客的到达或出发航班未被分配到固定登机口时，将会被忽略而不作考虑，这将导致求取总体最小流程时间的乘客数目是个随决策变量 y_{ik} 变化的变量，因此乘客的总体流程时间与航班分配决策紧密相关。

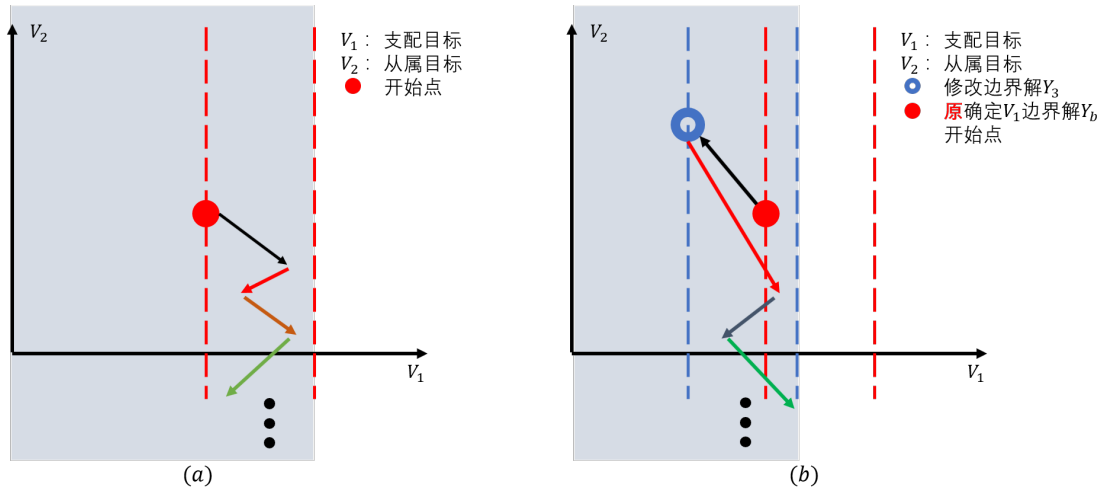
针对此问题，我们首先确定启发式搜索规则。进一步分析，对于同一架飞机而言，其到达航班类型是确定的，搭载的中转乘客数目是固定的，因此我们可以从与该飞机乘客关联的时间进行启发式规则的设计。此时，为了简化程序结构，根据多个目标优化的优先级，将到达航班对应的转机记录按照航班类型和飞机类型匹配完毕后，先对登机口进行排序；计算两种类型乘客的时间总和：一是，搭乘该飞机出发航班的乘客到达选择的登机口的时间总和，二是，从该飞机下机的乘客到达另一个出发航班且该航班的飞机已经抵达机场被安排了登机口的时间总和。这两个时间的累加作为时间权重。根据时间权重，选择总中转时间最小的登机口。同时，类似问题 1，在启发式搜索的基础上加入了概率和本文提出的搜索算法进行可行解中最优解的搜索。

不同于问题 1，问题 2 的模型具有三个目标，因此利用问题 1 的搜索算法结构进行求解的时候，需要对得到的解进行评价。为了简化评价方式，我们可先将问题近似为两目标优化问题，注意乘客的总体最小流程时间数量级为 1000 分钟级，而登机口数目的使用，最大则不超过 69。因此，我们可以将最小未分配航班数目与最少使用的登机口数目两者进行合并，记为 V_1，而乘客总的最短流程时间记为 V_2，其中 V_1 定义为支配目标，V_2 定义为从属目标。我们沿用问题 1 的求解结构求解问题 2 模型，并根据上述描述替换启发式规则；进行求解的时候，需要考虑如果筛选解以更新最佳决策线的问题。



(a)第i代　　(b)第$i+1$代

图 **8**　更新规则示意图



(a)　　(b)

图 **9**　解集目标函数值更新轨迹

如图8所示，对于（a），显然，黑色禁止符号的解 Y_1 比红色实心解 Y_b 要差，不予考虑，而绿色空心点，则属于问题 2 当前代数的 parato 解集，无法判断其与 Y_b 解的优劣；由于目标之间有从属关系，我们更关心的是横轴代价函数的减少，因此，在（a）中，右侧红色虚线限定的范围为以红色实心点为参考点的支配范围，灰色底纹区域以外的解将视为劣解，不用于更新用于变异的最佳决策线；设置允许裕度等价于允许牺牲一部分支配目标 V_1 的性能以换取更佳的 V_2 性能，而此时，支配目标仍然是占据支配地位的，这种刚设置的目的在于，允许搜索的解有更大范围的变动空间从而增大得到最优解的可能。

在该规则下，二维平面中表示的两目标函数轨迹将如图9中的（a）所示。图8（b）中示意了当下一代出现了一个以支配目标为参考的更佳的点 Y_3 时，最佳决策线的范围也随之移动，如灰色底纹所示。综合图8（a）和（b）中出现的情况，最终的解对应的目标函数轨迹将如图9（b）所示，不断往右上角及规定的裕度范

围内的右侧下方更新，最终不断收敛到最优解对应的目标函数值位置。本问题最优解搜索过程中，其目标值的分布图如10下所示，标号 1、2、3... 表示更新顺序，依次连接起来可获得更新的轨迹。

在本文中，以上思考过程对应的更新规则如以下伪代码所示：

```
[评价多目标解集的伪代码]
变量:
V_1 -> 当前迭代支配目标函数值
V_2 -> 当前迭代从属目标函数值
V1_best -> 当前最佳解的支配目标函数值
V2_best -> 支配目标下的从属目标函数值
过程:
00. 迭代中得到决策变量y_{ik}
01. V_1, V_2=评价决策变量y_{ik}
02. if V_1<V1_best Then
03.     更新V1_best
04.     更新V2_best
05.     更新最佳决策变量Y_best
06. else
07.  if V_1小于V_1加允许偏离所有解中最小的裕度v_{range}
08.    if V_2<V2_best Then
09.      更新V2_best
10.      更新最佳决策变量Y-best
11.    endif
12.  end if
13. end if
```

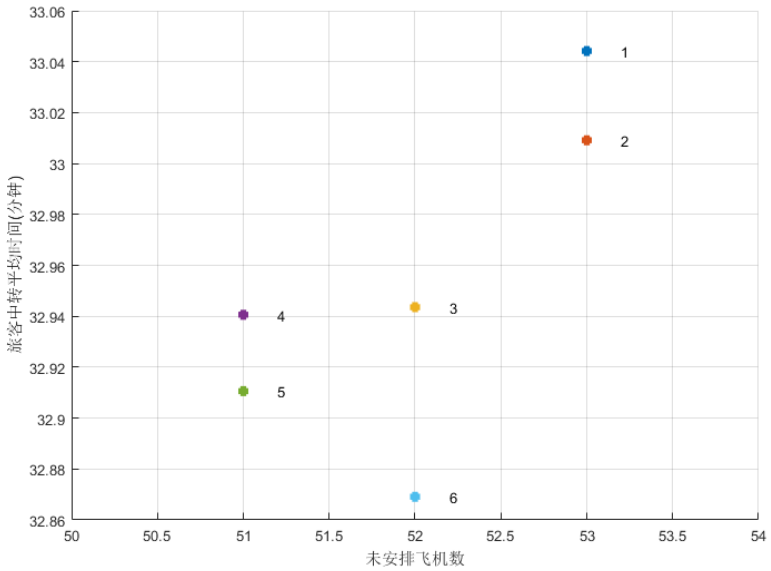


图 **10**　问题二两目标更新分布图

6.4 问题二模型的结果

1）结果：求解后可得有 253 架飞机的转接记录可被安排到合适的登机口，50 架飞机的转接记录未被安排登机口，69 个登机口共 67 个登机口被使用，剩余 2 个登机口 S29 和 S30 未被使用。具体每架飞机航班-登机口的分配结果如表4所示：（由于篇幅有限，按照到达时间先后排序后，展示前 20 架飞机成功分配到登机口的记录）

表 **4** 问题二中飞机航班-登机口的部分分配结果

飞机转场记录号	到达航班	出发航班	对应登机口	飞机转场记录号	到达航班	出发航班	对应登机口
PK208	NV847	NV690	T2	PK112	NV6253	NV316	T20
PK062	GN0523	GN0256	T10	PK117	NV6779	NV6738	T13
PK072	GN0497	GN0644	T11	PK129	NV319	NV846	S31
PK089	NV663	NV692	T3	PK131	NV673	NV320	S32
PK094	NV693	NV662	T4	PK136	NV6753	NV6358	T14
PK102	GN0209	GN0658	T12	PK144	NV621	NV322	T1
PK104	NV697	NV840	T26	PK142	GN0237	GN920	T21
PK106	NV6489	NV880	T25	PK145	NV6317	NV6540	T15
PK107	NV601	NV664	T27	PK147	NV6725	NV6724	T16
PK108	NV821	NV608	T28	PK148	NV6549	NV306	T7

将 69 个登机口接纳 253 架飞机转接记录用图11表示，图中纵坐标代表了 69 个登机口，每一段彩色部分为某一架飞机停在了对应的登机口上的示意图，空白部分则表示该登机口没有飞机停放，横坐标为以分钟为单位的时间轴。可以从图中明显看出，每一横轴的彩色部分没有重叠且有一定的间隔（大于等于 45 分钟），验证了结果的合理性。

2）结果：而 50 架飞机的转接记录未被安排登机口中，也展示前 10 架飞机转接记录如表5所示：

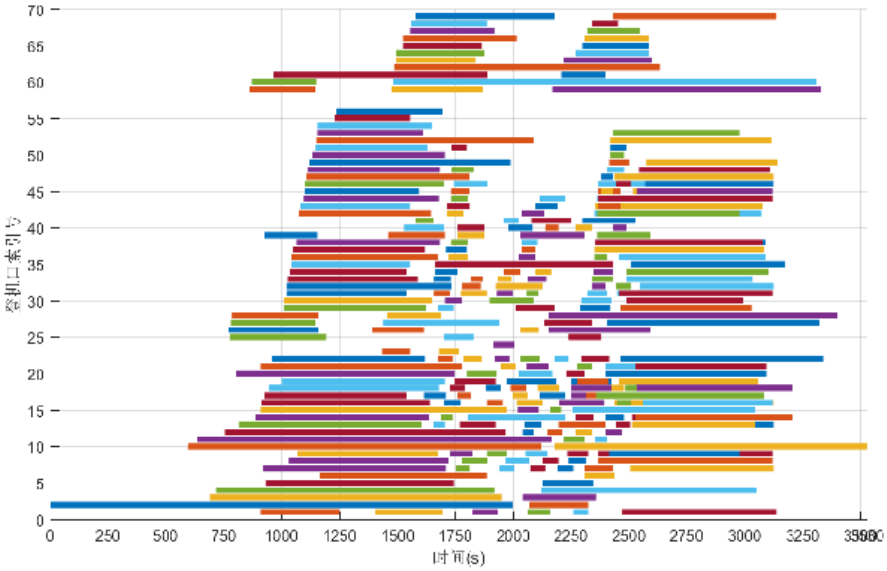


图 **11**　问题二 **69** 个登机口接纳 **253** 架飞机转接记录

表 **5**　问题二中未分配到登机口的部分飞机航班

飞机转场记录号	到达航班	出发航班	飞机转场记录号	到达航班	出发航班
PK165	NV6549	NV306	PK190	NV6549	NV306
PK177	NV6549	NV306	PK444	NV6549	NV306
PK183	NV6549	NV306	PK451	NV6549	NV306
PK186	NV6549	NV306	PK461	NV851	NV828
PK189	NV6549	NV306	PK464	NV689	NV318

3）结果：根据上述的问题分析，我们知道问题二中的三个优化子目标是无法同时达到最小值，因此原问题的最优解本身是难以寻找的。我们的策略就是将“未安排登机口的飞机尽可能少”作为基础性目标，然后才考虑最小化最短流程时间。在多任务优化中，首要目标往往不是越小越好，它可能只要足够小就已经能够让人满意，此时应该把优化的精力放在第二、第三和后面的子目标上，达到最大化满意度的权衡。

因此，我们在目标“未安排登机口的飞机尽可能少”设置了一个裕度，本问题中裕度设为 2，即如果再多两辆飞机进入临时机位也是可以容忍的，因此在最

小化旅客总体最短流程时间中，“未安排登机口的飞机尽可能少”目标的选择留有一定的裕度，得到的结果如表6所示。由于越多的飞机安排在临时机位，将会使得考虑的乘客数变少，从而对总体时间有影响，因此在本问题中，我们考虑了旅客换乘的平均最短流程时间。

表 **6**　问题二的三种结果对比表

未被安排登机口的飞机数量	被使用登机口的数量	旅客总体最短流程时间	旅客平均最短流程时间	中转失败数量比例
50（问题一的最优解）	67	65555	33.80	0
50	67	65015	33.54	0
51	67	61885	33.30	0
52	67	60865	33.31	0

对比表6中的第一第二行我们发现，引入最短流程时间目标后，总的流程时间和平均流程时间都有所减少。并且在能够容忍的裕度内，总的流程时间和平均流程时间也在减少。但由于牺牲“未安排登机口的飞机数量”的目标对流程时间的减少是几乎可以忽略不记的，因此本文针对问题二选择的解为表6中第二行对应的数值。

6.5　问题二结果的数据分析

1）给出成功分配到登机口的航班数量和比例，按宽、窄体机分别画线状图。

行程包含 20 号的所有转场记录的飞机一共有 303 架次，在 20 号当天共有 606 个航班。其中有 50 架，即 100 个航班没有分配到合适的登机口（安排在临时停机位），所以分配到登机口的航班数量为 506 个，成功分配的比例为 83.50%。将成功分配到每个宽型登机口和窄型登机口的航班数量和比例分别画图，如图12、13所示

2）给出 T 和 S 登机口的使用数目和被使用登机口的平均使用率（登机口占用时间比率），要求画线状图。

T 航站楼的登机口被使用了 28 个（总共 28 个），卫星厅 S 的登机口被使用了 39 个（总共 41 个），在 20 号一天每个登机口占用时间的比率如下图14所示，卫星厅 S29、S30 未被使用，即使用率为 0。

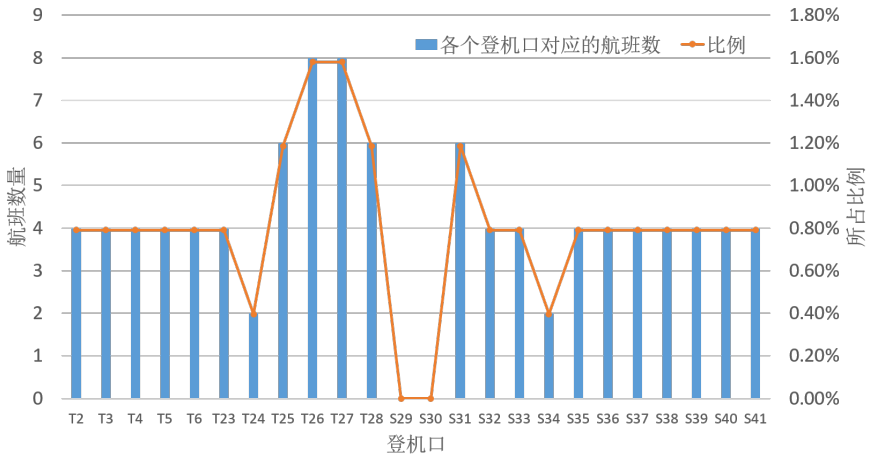


图 **12**　问题二中成功分配到宽型登机口的航班数量和比例图

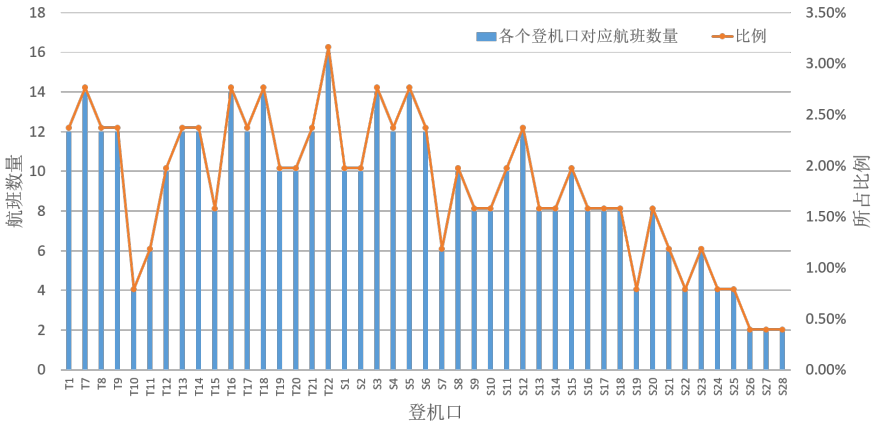


图 **13**　问题二中成功分配到窄型登机口的航班数量和比例图

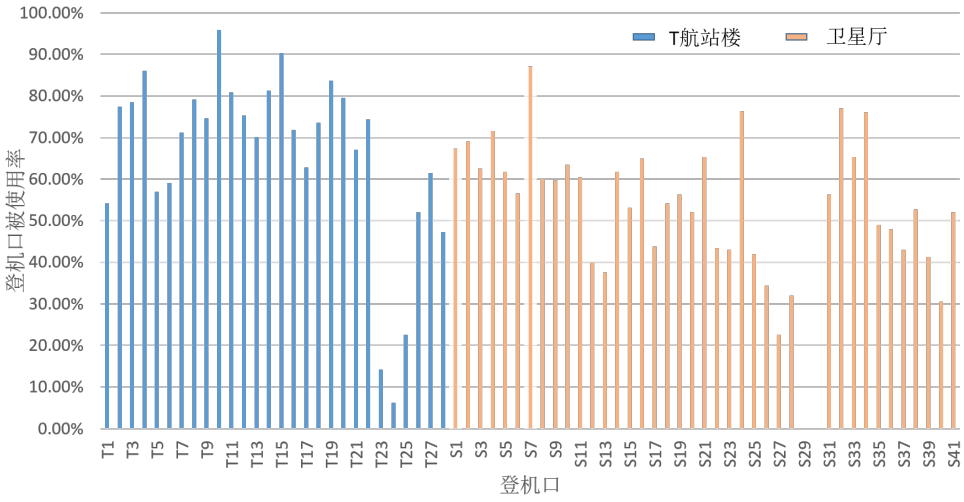


图 **14**　问题二中 **20** 号一天中每个登机口占用时间的比率图

3）给出换乘失败旅客数量和比率。

答：换乘失败旅客数量和比率均为 0。

4）总体旅客换乘时间分布图：换乘时间在 5 分钟以内的中转旅客比率，10 分钟以内的中转旅客比率，15 分钟以内的中转旅客比率，……

本文将对时间进行编号，如 0~5(包含 5 分钟) 为编号 1，5~10(包含 10 分钟) 为编号 2，以此类推。问题二中总体旅客换乘时间分布图如图15所示：

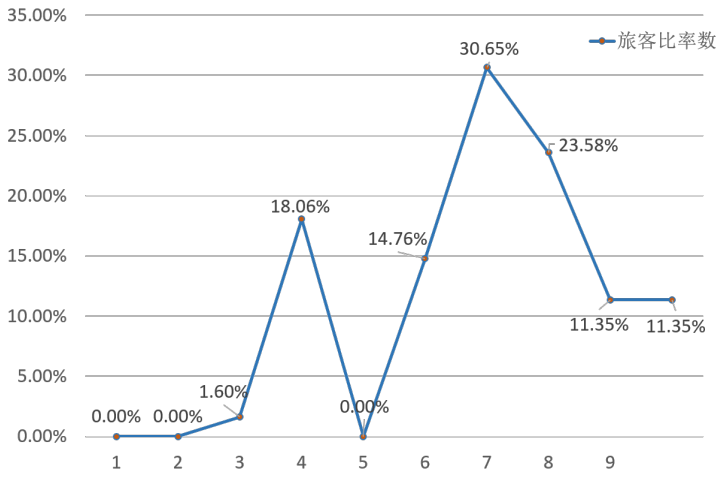


图 **15**　问题二中总体旅客换乘时间分布图

5）总体旅客换乘紧张度分布图：紧张度在 0.1 以内的中转旅客比率，0.2 以内的中转旅客比率，0.3 以内的中转旅客比率，……

由图16可知，紧张度在 0.1 以内中转旅客比率为 0.119，在 0.1 以上 0.2 以内中转旅客比率为 0.442，在 0.2 以上 0.3 以内中转旅客比率为 0.328，以此类推。

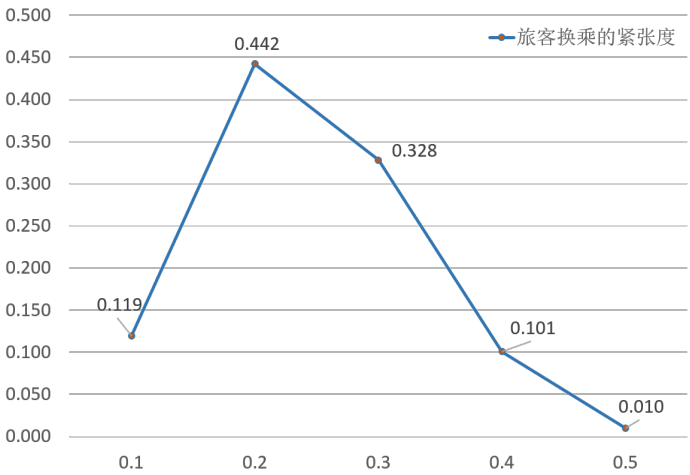


图 **16**　问题二中总体旅客换乘紧张度分布图

7. 问题三：中转旅客的换乘紧张度问题

7.1 问题三模型的建立

问题三是在问题二的基础上细化，考虑中转旅客的最短流程时间外，还需考虑捷运时间和行走时间，并将旅客换乘的紧张度作为目标函数的首要因素中，旅客换乘时间如公式（19），旅客换乘的紧张度如公式（20）。

$$T_p^{travellers} = T_p^{flow} + T_p^{mrt} + T_p^{wk} \tag{19}$$

$$\eta_p^{tensity} = \frac{T_p^{travellers}}{T_p^{ftrans}} \tag{20}$$

由假设三可知，计算旅客换乘总体紧张度时，仅考虑到达和出发航班均被分配了固定登机口的中转旅客，这部分被考虑的中转旅客有两种情况，一是换乘成功，即旅客的换乘时间（$T_p^{travellers}$）要小于乘坐的航班连接时间（T_p^{ftrans}）；二是换乘失败，即 $T_p^{travellers}$ 大于 T_p^{ftrans}，由假设四可知，其中转时间以 6 个小时 (360 分钟) 计入惩罚。本文用 $T_p^{tra\_success}$ 表示问题三中旅客 p 换乘是否成功。

$$T_p^{tra\_success} = (T_p^{ftrans} - T_p^{travellers}) \times \sum_{k=1}^{M} y_{p_a k} \times \sum_{k=1}^{M} y_{p_d k}, \quad \forall p \in \{1, ...n^{travel}\} \tag{21}$$

其中，$y_{p_a k}$ 中下标 p_a 表示旅客到达时乘坐的航班，下标 p_d 表示旅客出发时乘坐的航班。$y_{p_a k}$ 表示旅客到达时乘坐的 p_a 航班是否有停在第 k 登机口，若是则为 1，否则为 0。$y_{p_d k}$ 表示旅客出发时乘坐的 p_d 航班是否有停在第 k 登机口，若是则为 1，否则为 0。旅客换乘是否成功将产生不同的换乘时间：

$$T_p^{travel\_new} = \begin{cases} T_p^{travellers}, \ T_p^{tra\_success} > 0 \\ 0, \quad T_p^{tra\_success} = 0 \\ 360, \ T_p^{tra\_success} < 0 \end{cases} \tag{22}$$

其中，$T_p^{travel\_new}$ 时代表该旅客乘坐的航班没有被安排在登机口上，即这种情况下的旅客不在考虑范围内，即换乘时间记为 0。

则对应的旅客换乘紧张度为：

$$\eta_p^{ten\_new} = \frac{T_p^{travel\_new}}{T_p^{ftrans}} \tag{23}$$

第一步：确定模型的决策变量：

1) 飞机是否停到对应的登机口上的决策变量 y_{ik}：

$$y_{ik} = \begin{cases} 1 \\ 0 \end{cases} \tag{24}$$

与问题一相同，构成了一个 303×69 的决策矩阵。

2) 飞机是否停在了临时机位，即飞机是否没找到合适的登机口的决策变量 z_i：

$$z_i = \begin{cases} 1 \\ 0 \end{cases} \tag{25}$$

其中，1 代表了第 i 架飞机找不到合适的登机口，停在了临时停机位。0 则反之，第 i 架飞机没有停在临时机位。题目中假定临时停机位数量无限制，只关心第 i 架飞机是否停在临时机口，不关心停在哪个临时机口上，即 z_i 是一个 303×1 的向量。

第二步：确定模型的约束条件：

问题三是在问题二的基础上细化旅客换乘时间（包括了最短流程时间、捷运时间和行走时间），影响了目标函数，而约束条件仍然不变，即仍与问题一的相同为：

$$s.t. \begin{cases} \sum_{k=1}^{M} y_{ik} + z_i = 1, & \forall i \in \{1,...N\} \\ Ha_k \times Fa_i \times y_{ik} \geq 0, & \forall i \in \{1,...N\}, k \in \{1,...M\} \\ Hd_k \times Fd_i \times y_{ik} \geq 0, & \forall i \in \{1,...N\}, k \in \{1,...M\} \\ (G_k - P_i) \times y_{ik} = 0, & \forall i \in \{1,...N\}, k \in \{1,...M\} \\ (A_j - D_i - \alpha) \times y_{ik} \times y_{jk} \geq 0, & \forall\, i,j \in \{1,...N\}, i < j, k \in \{1,...M\} \end{cases}$$

第三步：确定模型的目标函数：由题意可知，需把中转旅客的换乘紧张度作为目标函数的首要因素，但基于问题一上，即目标条件为尽可能多地分配航班到合适的登机口的优先级为最高。

1）目标函数：最小化停在临时机口的飞机数量

$$min \sum_{i=1}^{N} z_i \tag{26}$$

2）子目标函数 1：最小化中转旅客的换乘紧张度

$$min \sum_{p=1}^{n^{travel}} (n_p^{ten\_new} \times Nt_p) \tag{27}$$

其中，$n_p^{tendity}$ 第 p 个旅客的换乘紧张度。

3）子目标函数 2：最小化被使用登机口的数量

$$min \sum_{k=1}^{M} \frac{\sum_{i=1}^{N} y_{ik}}{\sum_{i=1}^{N} y_{ik} + \varepsilon} \tag{28}$$

其中，ε 设定为极小数，本文设定为 0.0001，防止出现分母为 0 无意义的情况。

综上所述，问题二的模型为：

$$min\ \ F_{q3} = (70 \times f_{q31},\ f_{q32},\ f_{q33}) \tag{29}$$

$$s.t. \begin{cases} \sum_{k=1}^{M} y_{ik} + z_i = 1, & \forall i \in \{1, ...N\} \\ Ha_k \times Fa_i \times y_{ik} \geq 0, & \forall i \in \{1, ...N\}, k \in \{1, ...M\} \\ Hd_k \times Fd_i \times y_{ik} \geq 0, & \forall i \in \{1, ...N\}, k \in \{1, ...M\} \\ (G_k - P_i) \times y_{ik} = 0, & \forall i \in \{1, ...N\}, k \in \{1, ...M\} \\ (A_j - D_i - \alpha) \times y_{ik} \times y_{jk} \geq 0, & \forall\, i, j \in \{1, ...N\}, i < j, k \in \{1, ...M\} \end{cases}$$

其中，

$$\begin{cases} f_{321} = \sum_{i=1}^{N} z_i \\ f_{q32} = \sum_{p=1}^{n^{travel}} (n_p^{ten\_new} \times Nt_p) \\ f_{q33} = \sum_{k=1}^{M} \frac{\sum_{i=1}^{N} y_{ik}}{\sum_{i=1}^{N} y_{ik} + \varepsilon} \end{cases}$$

7.2 问题三模型的求解

问题三的求解过程与问题二的求解过程类似利用问题 1 中提及的搜索算法，修改启发式规则为计算当前航班累加下飞机旅客到达已分配登机口航班的紧张度及已在机场内即将要乘坐该飞机出发航班的旅客的紧张度，即仅考虑在已知信息内所能计算的所有紧张度。据此作为选择登机口的依据。利用问题 2 中的规则更新最佳决策线，不断迭代求解至收敛。同问题二，本文题求解得到的两个目标值更新轨迹如图17所示：

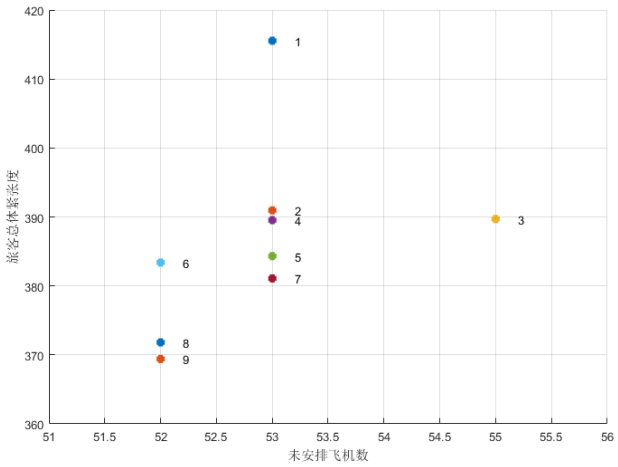


图 **17**　问题三两目标值更新轨迹

7.3 问题三模型的结果

1）结果：求解后可得有 253 架飞机的转接记录可被安排到合适的登机口，50 架飞机的转接记录未被安排登机口，69 个登机口共 67 个登机口被使用，剩余 2 个登机口 S29 和 S30 未被使用。具体每架飞机航班-登机口的分配结果如表7所示：（由于篇幅有限，按照到达时间先后排序后，展示前 20 架飞机成功分配到登机口的记录）

表 **7**　问题二中飞机航班-登机口的部分分配结果

飞机转场记录号	到达航班	出发航班	对应登机口	飞机转场记录号	到达航班	出发航班	对应登机口
PK208	NV847	NV690	T2	PK112	NV6253	NV316	T20
PK062	GN0523	GN0256	T10	PK117	NV6779	NV6738	T13
PK072	GN0497	GN0644	T11	PK129	NV319	NV846	S31
PK089	NV663	NV692	T3	PK131	NV673	NV320	S32
PK094	NV693	NV662	T4	PK136	NV6753	NV6358	T14
PK102	GN0209	GN0658	T12	PK144	NV621	NV322	T1
PK104	NV697	NV840	T26	PK142	GN0237	GN920	T21
PK106	NV6489	NV880	T25	PK145	NV6317	NV6540	T15
PK107	NV601	NV664	T27	PK147	NV6725	NV6724	T16
PK108	NV821	NV608	T28	PK148	NV6549	NV306	T7

将 69 个登机口接纳 253 架飞机转接记录用图18表示，图中纵坐标代表了 69 个登机口，每一段彩色部分为某一架飞机停在了对应的登机口上的示意图，空白部分则表示该登机口没有飞机停放，横坐标为以分钟为单位的时间轴。可以从图中明显看出，每一横轴的彩色部分没有重叠且有一定的间隔（大于等于 45 分钟），验证了结果的合理性。

2）结果：而 50 架飞机的转接记录未被安排登机口中，也展示前 10 架飞机转接记录如表8所示：

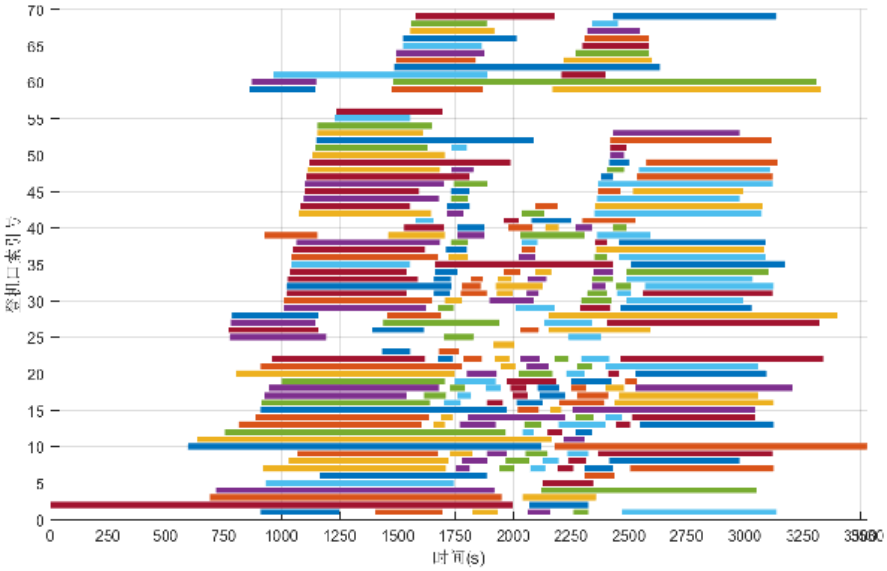


图 **18**　问题三 **69** 个登机口接纳 **253** 架飞机转接记录

表 **8**　问题二中未分配到登机口的部分飞机航班

飞机转场记录号	到达航班	出发航班	飞机转场记录号	到达航班	出发航班
PK165	NV6549	NV306	PK190	NV6549	NV306
PK177	NV6549	NV306	PK444	NV6549	NV306
PK183	NV6549	NV306	PK451	NV6549	NV306
PK186	NV6549	NV306	PK461	NV851	NV828
PK189	NV6549	NV306	PK464	NV689	NV318

3）结果：与问题二的结果分析相类似，我们对基础性的目标“未安排登机口的飞机尽可能少”设置了一个裕度 2，来达到最大化满意度的权衡。所得到的针对本问题的结果如表9所示。根据表中的第一第二行可以看出，针对最小化乘客紧张度的目标函数的引入能够使得在保证“未安排登机口的飞机尽可能少”的基础上乘客登机的紧张度最小。在裕度允许的范围内，裕度放宽越多，紧张度也可以求解的却小，因此适当牺牲较好目标的损失可以让其它目标变优到一个整体目标让人满意的效果。

表 **9**　问题三的三种结果对比表

未被安排登机口的飞机数量	被使用登机口的数量	旅客总体换乘时间	旅客换乘总体紧张度	中转失败数量比例
50（问题一的最优解）	67	112646	400.03	0
50	67	111333	396.29	0
51	67	108735	388.64	0
52	67	105156	371.49	0

7.4　问题三结果的数据分析

1）给出成功分配到登机口的航班数量和比例，按宽、窄体机分别画线状图。

行程包含 20 号的所有转场记录的飞机一共有 303 架次，在 20 号当天共有 606 个航班。其中有 50 架，即 100 个航班没有分配到合适的登机口（安排在临时停机位），所以分配到登机口的航班数量为 506 个，成功分配的比例为 83.50%。将成功分配到每个宽型登机口和窄型登机口的航班数量和比例分别画图，如图19、20所示。

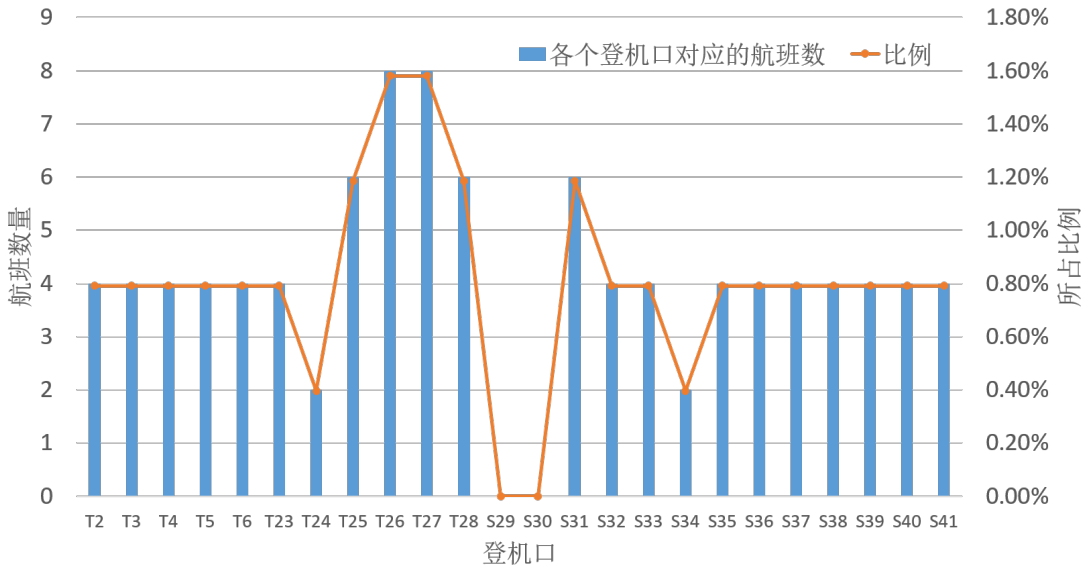


图 **19**　问题三中成功分配到宽型登机口的航班数量和比例图

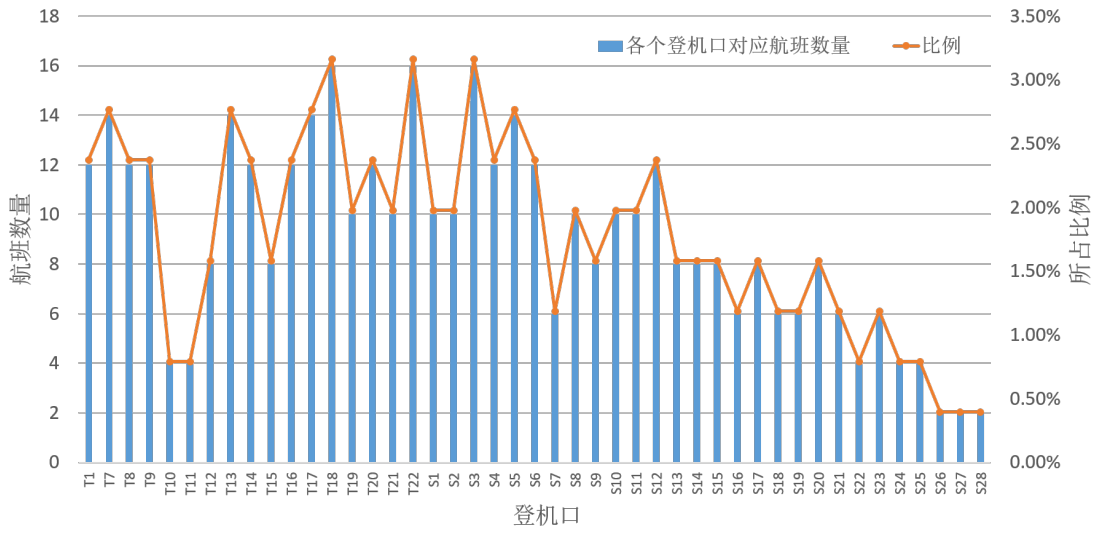


图 **20**　问题三中成功分配到窄型登机口的航班数量和比例图

2）给出 T 和 S 登机口的使用数目和被使用登机口的平均使用率（登机口占用时间比率），要求画线状图。

T 航站楼的登机口被使用了 28 个（总共 28 个），卫星厅 S 的登机口被使用了 39 个（总共 41 个），在 20 号一天每个登机口占用时间的比率如下图21所示，卫星厅 S29、S30 未被使用，即使用率为 0。

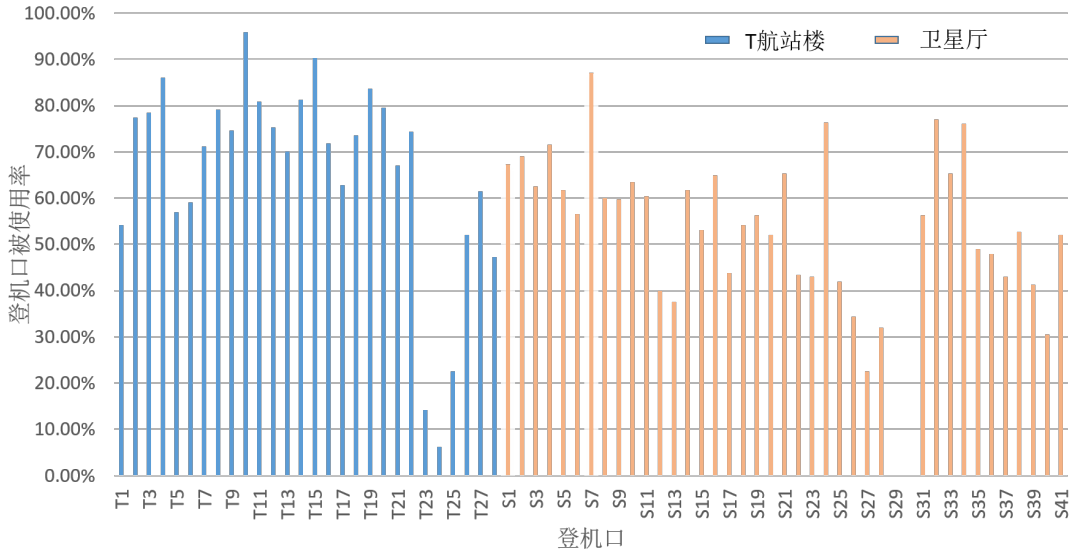


图 **21**　问题三中 **20** 号一天中每个登机口占用时间的比率图

3）给出换乘失败旅客数量和比率。

答：换乘失败旅客数量和比率均为 0。

4）总体旅客换乘时间分布图：换乘时间在 5 分钟以内的中转旅客比率，10 分钟以内的中转旅客比率，15 分钟以内的中转旅客比率，……

同问题二，对时间进行编号，如 0~5(包含 5 分钟) 为编号 1，5~10(包含 10 分钟) 为编号 2，以此类推。问题二中总体旅客换乘时间分布图如图22所示：

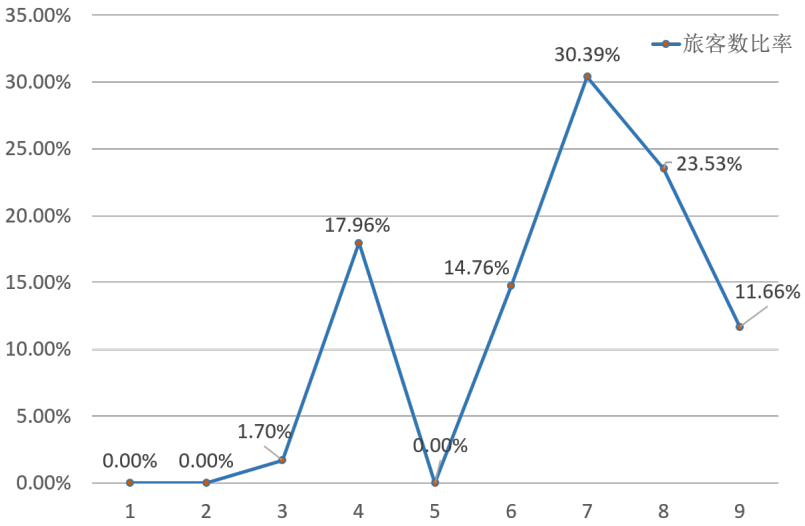


图 **22**　问题三中总体旅客换乘时间分布图

5）总体旅客换乘紧张度分布图：紧张度在 0.1 以内的中转旅客比率，0.2 以内的中转旅客比率，0.3 以内的中转旅客比率，……

由下图23可知，紧张度在 0.1 以内的中转旅客比率为 0.121，在 0.1 以上 0.2 以内的中转旅客比率为 0.431，在 0.2 以上 0.3 以内的中转旅客比率为 0.338，以此类推。

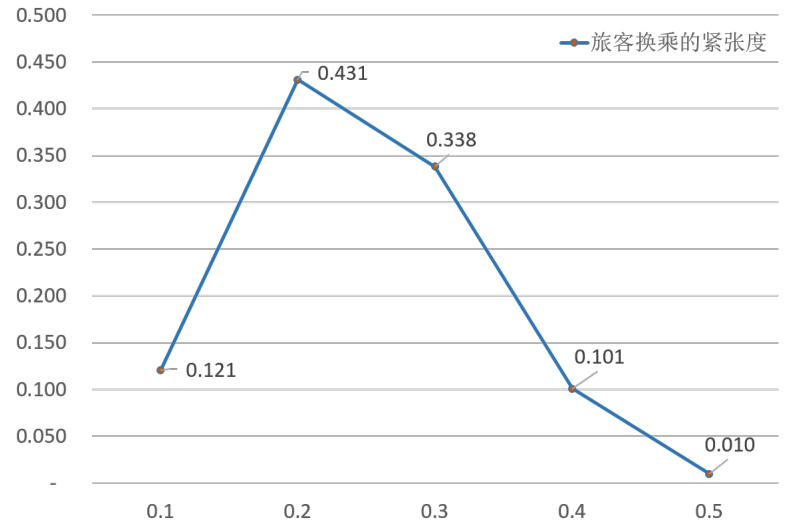


图 **23**　问题三中总体旅客换乘紧张度分布图

8. 模型的讨论与评价

8.1 模型的优点

数学模型

本文采用的数学模型能够清晰简明地反映我们实际问题地目标以及各种约束，它主要包含以下优点：

- 决策变量能够针对停机位分配问题，模型思路清晰，约束方程简明，中间变量尽可能少，有利于加速模型的求解。
- 目标函数定义巧妙。如最小化登机口目标函数将判别问题转化成一个数值计算问题；第二第三问的目标函数抛开传统固定死板的模式，引入可容忍裕度，使得我们的模型能够更好地搜索出一个整体上让人满意地结果。

算法模型

本文没有采用任何的算法工具箱，算法的思想是针对本问题设计的。算法模型的主要优点为：

- 先验知识给算法搜索提供方向。由于本问题解空间非常庞大，约束非常多，因此本文根据很多先验知识取指定搜索方向和大致策略。（贪婪算法思想）
- 随机概率为算法全局搜索提供可能。明确的策略只会让算法陷入局部最优解，而无法找到最优解甚至是更优解。引入概率随机，对算法搜索路径进行变异和随机选择，可以使规则实施具有灵活性，能够衍生出更多更好的可能。（改进遗传算法思想）
- 路径繁衍为算法最优路径选择提供指导。明确的策略无法产生创造性的策略，而随机概率搜索难以较好的收敛和稳定。而引入最优路径的繁衍过程可以保证算法能够将好的基因（路径）繁衍到下一代，在此基础上进行概率搜索将更容易找到最优解。
- 在寻找最优解和搜索速度上寻找权衡，能够快速搜索到一个满意的较好的解。

8.2 模型的改进

本文采用的算法既不是纯粹的贪心算法，也不是纯粹的遗传算法，但同时结合了两种算法的优势并针对本次问题进行的一次改进。主要改进如下：

- 在贪婪算法上引入随机概率决策。
- 将航班登机口分配问题转化成路径规划模型，引入变异点，此处变异方式有别于传统遗传算法的变异。
- 将启发式算法和优势路径繁衍策略相结合，即将贪婪算法与遗传算法相结合，设计出一套针对本问题的优化算法。

本文建立的航班-登机口匹配问题模型，并最小化旅客换乘时间因素，对于实际中航空公司对航班与登机口的选择上具有一定的实际用途，可在一定程度上降低由于航班没有登机口停放带来的成本。而在当下航空行业的飞速发展，飞机已经广泛成为人们出行的交通方式选择之一，对于旅客的换乘时间安排更需着重考虑，为广大旅客提供良好的出行体验，因此，本模型实用性较强，适用面广。

9. 参考文献

[1] $Hemchand\,Kochukuttan\,and\,Sergey\,Shebalov$, $New\,Gate\,Planning\,Optimizer\,Perfects\,\,Gate\,\,Assignment\,\,Process$, $Ascend\,\,Magazine$, $https$: $//www.sabreairlinesolutions.com/pdfs/Plan_{A}head.pdf$

[2] 卫东选, 刘长有. 机场停机位分配问题研究 [J]. 交通运输工程与信息学报, 2009, 7(1): 57 - 63.

[3] 卫东选. 基于改进遗传算法的机场停机位分配问题研究 [D]. 中国民用航空学院中国民航大学, 2006.

附录 A　MATLAB 源程序

程序的执行过程已在前文中的伪代码中提及，附件的代码中对相关语句已充分注释，在此不再赘述。程序相关文件内容给出如下：

【Main.m】

*problem*1_*main.m*、*problem*2_*main.m* 、*problem*3_*main.m* 分别对应问题 1,2,3 的求解过程。

```
%%% 引入变异，引入随机停进临时机位的策略
% Ftrans  -> 转场记录
% Travellers -> 乘客转机信息（用于查表使用）
% GatesInfo -> 登机口相关信息
[ FTrans,Tickets,GatesInfo]=data_read('filtered_data.xlsx');
obj1 = 0;                % 目标函数：最小化停止临时机位的数量
obj2 = 0;                % 第二问子目标1
obj3 = 0;                % 第三问子目标1
point = 304;             %% 初始变异的点，即第一次不进行变异
point0 = 0;              %% 变异区域起点，确定随机规则选择
point1 = 0;              %% 变异区域终点，确定随机规则选择
AveTransTime_obj1 = 0;   %记录满足目标一条件下最好的平均流程时间
Sum_Tense_obj1 = 0;      %记录满足目标一条件下最好的紧张度
load('problem3_5.mat')   %加载上一次训练的模型
Y_IKparent = Y_IK1{6}; %%Y_IK1;
Y_IK1 = {};              % 每次目标函数1减小，保存决策变量
Y_IK1_1 = {};            % 每次问题2的子目标函数1减小，保存决策变量
Y_IK1_2 = {};            % 每次问题3的子目标函数2减小，保存决策变量
Y_IK2 = {};              % 在一定的裕度下，每次问题2的子目标函数1减小，保存决策变量
Y_IK3 = {};              % 在一定的裕度下，每次问题3的子目标函数1减小，保存决策变量
YuduV2 = [100,100,100]; % 问题2中，满足2裕度区间的滑窗，专门保存滑窗的最小值
YuduV3 = [1000,1000,1000]; % 问题3中，满足2裕度区间的滑窗，专门保存滑窗的最小值
position1 = {};          % 保存目标1每次变好的坐标值->
    （停在临时机位的数量，登机口使用数量）
position2 = {};          % 保存目标2每次变好的坐标值->
    （停在临时机位的数量，换乘平均最小流程时间）
position3 = {};          % 保存目标3每次变好的坐标值->
    （停在临时机位的数量，换乘平均最小换乘时间）
Use_port = 70;           % 使用登机口数初始化为70
obj1_update_flag = 0;    %
    本次迭代是否有目标1的更新，0表示没有，大于1则有更新，更新大小为对应数值
N=length(FTrans);        % 专场记录总数
K=length(GatesInfo)+1;   % 包括临时机场
for epoch = 1:200000
  Y_ik=zeros(N,K);      % 决策变量
  Gates_available=zeros(K,1); % 登机口可用时间
  num_unassign=0;       % 没分配到登机口的飞机数量
```

```
if epoch>1
  point = round(rand(1,1)*303); %%
      生成变异节点，第一次迭代不需要生成变异节点。变异节点前的路径沿用父代的路径
end
point0 = point + round((303-point).*rand(1,1)); %
    概率决策起点。没有概率决策的部分沿用固定规则决策当前航班
point1 = point + round((303-point).*rand(1,1)); % 概率决策终点
if point1<=point0
  orientation = 0;                          %
      概率决策段的方向。方向0，选择point0和point1段两侧的段进行概率决策
else
  orientation = 1;                          %
      概率决策段的方向。方向1，选择point0到point1段之间进行概率决策
end
for ft=1:N                                   % 对每个转场记录进行分配
  ftrans=FTrans{ft};
  if ft < point                             % 小于变异节点的所有节点沿用附带路径
    choose_gate = find(Y_IKparent(ft,:)==1);
    Y_ik(ft,choose_gate)=1;
    Gates_available(choose_gate)=ftrans.leaveTime+2700; % 离开时间+45min
    FTrans{ft}.arriveSTidx=choose_gate;
    else                                      % 大于等于变异节点
    ok_gates=[];
    gate_match=false;
    for gidx=1:K-1 % 匹配登机口类型
    gate=GatesInfo{gidx};
    if (ftrans.arriveType*gate.arriveType>=0 &&
        ftrans.leaveType*gate.leaveType>=0 ... %
        判断是否满足登机口宽窄和国内国际型号要求
    && ftrans.planeType==gate.planeType)
      gate_match=true;
    if ftrans.arriveTime>=Gates_available(gidx) % 时间约束检测
      ok_gates=[ok_gates; gidx];            % 满足约束的所有登机口
    end
  end
end
if isempty(ok_gates)                          % 分配至临时停机场
  Y_ik(ft,K)=1;
  num_unassign = num_unassign + 1;
else
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % 贪心算法做决策，可以加算法
  ok_gates=gate_priority(GatesInfo,ok_gates,ftrans); % 对可用登机口排权重
  if orientation == 1                         % 方向为正向
    if (ft >= point0 && ft <= point1)|| ft == point %
        在概率决策段或者变异点进行概率决策
      if rand() > 0.07                        % 93% 不放入临时机位
```

```
        p0 = 1.0;     %% 概率
        p1 = 0;
        P = [];
        for p_i = 1:length(ok_gates) - 1  % 概率决策的概率分布
          p1 = 0.4 * p0;
          P = [P,p1];
          p0 = 1-p1;
        end
        P = [P,1-sum(P)];
        gambel=0; r = rand();              % 轮盘筛选
        for choose_idx=1:length(P)
          if r>gambel && r<gambel+P(choose_idx)
            break;
          else
            gambel = gambel + P(choose_idx);
          end
        end
      else                                 % 7% 随机放入临时机位
        choose_idx = length(ok_gates)+1;
      end
    else
      choose_idx = 1;
    end
  elseif orientation==0                      %% 如果变异反向
    if (ft < point0 || ft > point1) || ft == point
      if rand()>0.07                       % 93% 不放入临时机位
        p0 = 1.0;
        p1 = 0;
        P = [];
        for p_i = 1:length(ok_gates) - 1  % 概率决策的概率分布
          p1 = 0.4 * p0;
          P = [P,p1];
          p0 = 1-p1;
        end
        P = [P,1-sum(P)];
        rand('seed',sum(100*clock));
        gambel=0; r = rand();              % 轮盘筛选
        for choose_idx=1:length(P)
          if r>gambel && r<gambel+P(choose_idx)
            break;
          else
            gambel = gambel + P(choose_idx);
          end
        end
      else                                 % 7% 随机放入临时机位
        choose_idx = length(ok_gates)+1;
      end
```

```
      else
        choose_idx = 1;
      end
    end
    ok_gates = [ok_gates;70];
    choose_gate = ok_gates(choose_idx); % 选择停放的登机口
    Y_ik(ft,choose_gate)=1;       % 更新决策变量
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % 更新gates的可用时间
    Gates_available(choose_gate)=ftrans.leaveTime+2700; % 离开时间+45min
    FTrans{ft}.arriveSTidx=choose_gate;
    end
end
end
% 代价函数计算
[cost1,cost2,cost3,ave_transtime1,ave_transtime2,sum_Tense2,use_port,unassign,Log]
    = cost_calc2(FTrans,Tickets,GatesInfo,Y_ik,0);

if mod(epoch,100) == 0
  disp(['%%%%%%%%%%%%%epoch%%%%%%%%%%:',num2str(epoch),'
      ,''sum_Tense2',num2str(sum_Tense2)]);
end
if epoch == 1                      % 第一次迭代进行各种初始化
  obj1 = unassign;
  Y_IK1 = [Y_IK1,Y_ik];
  Y_IK1_1 = [Y_IK1_1,Y_ik];
  Y_IK1_2 = [Y_IK1_2,Y_ik];
  Y_IK2 = [Y_IK2,Y_ik];
  Y_IK3 = [Y_IK3,Y_ik];
  if isempty(Y_IKparent)          % 如果第一次没有导入Y_IKparent 预训练模型
    Y_IKparent = Y_ik;
  end
  disp(['目标函数1: ',num2str(obj1),' ', '目标函数2',num2str(obj2), '
      ','目标函数3',num2str(obj3)])
  disp(['使用固定登机口数目: ',num2str(use_port)]);
  disp(['未被分配的飞机数目: ',num2str(unassign)]);
  disp(['平均转移时间(problem2): ',num2str(ave_transtime1)]);
  disp(['平均换乘紧张度(problem3): ',num2str(sum_Tense2)]);
  disp('%%%%%%%%%%%%%%%%%%');
else
  if unassign<obj1                          % 如果本次搜索使得问题1的目标函数变小
    obj1 = unassign;
    obj1_update_flag = obj1 - unassign; % 目标函数更新变化的值
    AveTransTime_obj1 = ave_transtime1;
    Sum_Tense_obj1 = sum_Tense2;
    Y_IK1 = [Y_IK1,Y_ik];                % 保存决策变量
    Y_IK1_1 = [Y_IK1_1,Y_ik];
```

```
    Y_IK1_2 = [Y_IK1_2,Y_ik];
    Y_IKparent = Y_ik;
    disp(['目标函数1: ',num2str(obj1) ,'
        ','使用登机口数: ',num2str(Use_port)])
    disp(['使用固定登机口数目: ',num2str(use_port)]);
    disp(['未被分配的飞机数目: ',num2str(unassign)]);
    disp(['平均转移时间(problem2): ',num2str(ave_transtime1)]);
    disp(['平均换乘紧张度(problem3): ',num2str(sum_Tense2)]);
    disp('%%%%%%%%%%%%%%%%%%%');
  end
  if unassign==obj1 %
      目标函数等于最优目标，判断问题2，3中的子目标是否变优，以及登机口使用数量是否减少。
                  % 有的话需要更新保存决策变量
    if ave_transtime1 < AveTransTime_obj1
      AveTransTime_obj1 = ave_transtime1;
      Y_IKparent = Y_ik;
      Y_IK1_1 = [Y_IK1_1,Y_ik];
      disp(['目标函数1-1: ',num2str(obj1)])
      disp(['使用固定登机口数目: ',num2str(use_port)]);
      disp(['未被分配的飞机数目: ',num2str(unassign)]);
      disp(['平均转移时间(problem2): ',num2str(ave_transtime1)]);
      disp(['平均换乘紧张度(problem3): ',num2str(sum_Tense2)]);
      disp('%%%%%%%%%%%%%%%%%%%');
    end
    if sum_Tense2 < Sum_Tense_obj1
      Sum_Tense_obj1 = sum_Tense2;
      Y_IKparent = Y_ik;
      Y_IK1_2 = [Y_IK1_2,Y_ik];
      disp(['目标函数1-2: ',num2str(obj1)])
      disp(['使用固定登机口数目: ',num2str(use_port)]);
      disp(['未被分配的飞机数目: ',num2str(unassign)]);
      disp(['平均转移时间(problem2): ',num2str(ave_transtime1)]);
      disp(['平均换乘紧张度(problem3): ',num2str(sum_Tense2)]);
      disp('%%%%%%%%%%%%%%%%%%%');
    end
    if(use_port<Use_port)
      Use_port = use_port;
      Y_IK1 = [Y_IK1,Y_ik];
      position1 = [position1, [unassign,use_port]];
      Y_IKparent = Y_ik;
      disp(['目标函数1: ',num2str(obj1),'
          ','使用登机口数: ',num2str(Use_port)])
      disp(['使用固定登机口数目: ',num2str(use_port)]);
      disp(['未被分配的飞机数目: ',num2str(unassign)]);
      disp(['平均转移时间(problem2): ',num2str(ave_transtime1)]);
      disp(['平均换乘紧张度(problem3): ',num2str(sum_Tense2)]);
      disp('%%%%%%%%%%%%%%%%%%%');
```

```
    end
end
if obj1_update_flag ~= 0 % 目标函数obj1 变好，需要更新裕度的滑窗最小值
    if obj1_update_flag == 1 %
        排在临时机位的数量减少1，滑窗往左移动一位，相当于数值往右移动一位
        YuduV2(3) = YuduV2(2);
        YuduV2(2) = YuduV2(1);
        YuduV2(1) = 100;
        YuduV3(3) = YuduV3(2);
        YuduV3(2) = YuduV3(1);
        YuduV3(1) = 1000;
    elseif obj1_update_flag == 2 %
        排在临时机位的数量减少2，滑窗往左移动两位，相当于数值往右移动两位
        YuduV2(3) = YuduV2(1);
        YuduV2(2) = 100;
        YuduV2(1) = 100;
        YuduV3(3) = YuduV3(1);
        YuduV3(2) = 1000;
        YuduV3(1) = 1000;
    else                        %
        滑窗往左移动三位，滑出原来区域，因此从新初始化滑窗最小值
        YuduV2 = [100,100,100];
        YuduV3 = [1000,1000,1000];
    end
    obj1_update_flag = 0;
end
if unassign <= obj1 + 2      % 目标函数只需要满足一个裕度2
    if ave_transtime1 < min(YuduV2) %
        如果平均流程时间减少，则问题2的子目标1变好，更新和保存决策变量
        idx = unassign - obj1 + 1;
        YuduV2(idx) = ave_transtime1; % 更新YuduV2
        position2 = [position2, [unassign,ave_transtime1]];
        Y_IK2 = [Y_IK2,Y_ik];
        disp(['目标函数2: ',num2str(ave_transtime1)])
        disp(['使用固定登机口数目: ',num2str(use_port)]);
        disp(['未被分配的飞机数目: ',num2str(unassign)]);
        disp(['平均转移时间(problem2): ',num2str(ave_transtime1)]);
        disp(['平均换乘紧张度(problem3): ',num2str(sum_Tense2)]);
        disp('%%%%%%%%%%%%%%%%%%');
    end
    if sum_Tense2 < min(YuduV3)  %
        如果总的紧张度减少，则问题3的子目标1变好，更新和保存决策变量
        idx = unassign - obj1 + 1;
        YuduV3(idx) = sum_Tense2; %% 更新YuduV2
        position3 = [position3, [unassign,sum_Tense2]];
        Y_IK3 = [Y_IK3,Y_ik];
        disp(['目标函数3: ',num2str(sum_Tense2)])
```

```
            disp(['使用固定登机口数目: ',num2str(use_port)]);
            disp(['未被分配的飞机数目: ',num2str(unassign)]);
            disp(['平均转移时间(problem2): ',num2str(ave_transtime1)]);
            disp(['平均换乘紧张度(problem3): ',num2str(sum_Tense2)]);
            disp('%%%%%%%%%%%%%%%%%%');
        end
    end
  end
end
```

【$data\_read.m$】

功能：文件读取经过 excel 软件筛选过后的文件 $filtered\_data.xlsx$，返回细胞体结构的：

1）FTrans　-> flight transfer 飞行换乘记录

2）Tickets　- 乘客订单

3）GatesInfo -> 登机口的相关信息

以上三个细胞体内的每个元素分别是结构相同的结构体，结构体中包含的是 $filtered\_data.xlsx$ 文件中的对应 sheet 中的被提取过后的每一行的有效信息，在读取的过程中对字符串进行了编码。

1）FTrans 细胞体内包含了 id（飞机转场记录号）、arriveTime（到达的时间，用 Unix 格式）、arriveID（到达航班号）、arriveType（到达航班类型）、leaveTime（出发时间，用 Unix 格式）、leaveId（出发航班号）、leaveType（出发航班类型）、planeType（飞机的类型）、intervalTime（航班时间间隔）、passagerOut（乘坐航班离开的旅客）、passagerIn（乘坐航班到达的旅客）、arriveSTidx

2）FTrans 细胞体内包含了 id（旅客记录号）、num（每个 id 对应的乘客数）、$FTrans\_arriveIdx$（到达航班）、arriveType（出发航班）、$FTrans\_arriveIdx$（到达航班对应的转场记录号）、$FTrans\_leaveIdx$（出发航班对应的转场记录号）、leaveType（离开类型）、connectTime（两个航班连接时间）

3）GatesInfo 细胞体内包含了 gateId（登机口）、arriveType（到达类型）、leaveType（出发类型）、planeType（机体类别）

```
% 载入文件,  已在excel中筛选了20号到达和出发的航班和旅客信息
[~,~,Pucks] = xlsread(filename,1); Pucks(1,:) =[]; % 删除表头
[~,~,Tckes] = xlsread(filename,2); Tckes(1,:) =[];
[~,~,Gates] = xlsread(filename,3); Gates(1,:) =[];

% 载入数据
FTrans = cell(length(Pucks),1); % 1. 航班数据的预处理,Flight Transfer
for fidx=1:length(Pucks)
```

```
    ftrans.id = Pucks{fidx,1};
    ftrans.arriveTime = Pucks{fidx,13};
    ftrans.arriveId = Pucks{fidx,4};
    ftrans.arriveType = flightType(Pucks{fidx,5});
    ftrans.leaveTime = Pucks{fidx,14};
    ftrans.leaveId = Pucks{fidx,9};
    ftrans.leaveType = flightType(Pucks{fidx,10});
    ftrans.planeType = planeType(Pucks{fidx,6}); % 得转为为W1或NO
    arrivetime = max(ftrans.arriveTime,1516406400);
    leavetime = min(ftrans.leaveTime,1516492800);
    ftrans.intervalTime = leavetime - arrivetime;
    ftrans.passagerOut = {}; % 航班到达时,离开飞机的旅客（预处理时更新）
    ftrans.passagerIn = {}; % 航班离开时,进入飞机的旅客（在决策中更新）
    ftrans.arriveSTidx = []; % 转场记录对应飞机所在的登机口位置，若为空则表示没有到达
    FTrans{fidx} = ftrans;
end

Tickets=cell(length(Tckes),1);
for tidx = 1:length(Tckes)
    %       passager.id = Tckes{tidx,1}; % 同程号id
    passager.num = Tckes{tidx,2}; % 同程号对应的乘客数目
    passager.Tidx=get_FtranIdx(FTrans,Tckes{tidx,15}); %
        到达机场的乘客想要转乘的航班对应的序列(Transfer index)
    fidx=get_FtranIdx(FTrans,Tckes{tidx,10}); % 到达航班对应的转场飞机记录号索引
    FTrans{fidx}.passagerOut=[FTrans{fidx}.passagerOut;passager]; %
        记录乘客到达的信息
    tickets.id =Tckes{tidx,1}; % 记录旅客票据信息用于后期计算行走时间等
    tickets.num=Tckes{tidx,2};
    tickets.FTrans_arriveIdx= get_FtranIdx(FTrans,Tckes{tidx,10});%
        到达航班对应的转场记录索引号
    tickets.arriveType=FTrans{fidx}.arriveType;    % 到达航班类型
    tickets.FTrans_leaveIdx = get_FtranIdx(FTrans,Tckes{tidx,15});%
        出发航班对应的转场记录索引号
    tickets.leaveType = FTrans{tickets.FTrans_leaveIdx}.leaveType; %
        出发航班类型
    tickets.connectTime =
        FTrans{tickets.FTrans_leaveIdx}.leaveTime-FTrans{fidx}.arriveTime;
        % 转乘乘客航班连接时间
    Tickets{tidx}=tickets;
end

GatesInfo=cell(length(Gates),1); % 3. 航站楼信息数据预处理
areaStr={'North','Center','South','East'};
for gidx = 1:length(Gates)
    gate.gateId = Gates{gidx,1};
    if strcmp(Gates{gidx,2},'T') % S类型航站楼 TorS=1, T类型航站楼 TorS=0
        gate.TorS = 0;
```

```
    else
        gate.TorS = 1;
    end
    for dir=1:length(areaStr) % 北，中，南，东方位依次记为1,2,3,4
      if strcmp(Gates{gidx,3},areaStr{dir})
        gate.areaIdx=dir; break;
      end
    end
    gate.arriveType= flightType(Gates{gidx,4});
    gate.leaveType = flightType(Gates{gidx,5});
    gate.planeType = planeType(Gates{gidx,6});
    GatesInfo{gidx} = gate;
  end

end
```

【*cost_calc.m*】

功能：计算当决策变量 Y_ik 确定好后，对应问题的考虑因素计算在内的代价函数值。如果最后一个输入变量 isLog 设置为 true，将在命令行中输出决策结果的信息。

```
% 输入相关的信息和决策变量
% 输出问题123的代价函数值，乘客票据对应的可用转乘时间
% isLog表示是否记录数据，如旅客对应的实际中转时间(false 或者 ture)
Y_k=sum(Y_ik,1)./(sum(Y_ik,1)+0.0001); %
    sum(Y_k)表示每个机场一天当中分配的航班数
K=length(GatesInfo)+1;
cost1=sum(Y_k)+70*sum(Y_ik(:,K));
sum_costTime_1 =0; sum_costTime_2 =0; % 旅客中转的总体时间(单位：分钟)
num_transFail_1=0; num_transFail_2=0; % 问题2 和 问题3对应的转机失败的订单数
num_scheduleSucess=0;
if isLog
  LogTime =[]; LogTense=[]; % 乘客换乘紧张度
end
for tidx=1:length(Tickets)
  tickets=Tickets{tidx};
  arrfIdx=tickets.FTrans_arriveIdx;
  leafIdx=tickets.FTrans_leaveIdx;
  arr_gate=FTrans{arrfIdx}.arriveSTidx;
  lea_gate=FTrans{leafIdx}.arriveSTidx;
  if ~isempty(arr_gate) && ~isempty(lea_gate) % 航班均会到达机场
  %         disp(['sucess arrange: from ',num2str(arr_gate),' to:
      ',num2str(lea_gate)]);
    num_scheduleSucess=num_scheduleSucess+tickets.num; % 被安排到的旅客
    [T_pc,T_ts,T_wk]=get_PassagerTime(
        GatesInfo,tickets.arriveType,arr_gate,tickets.leaveType,lea_gate);
```

```
      time_connect=tickets.connectTime/60;
      if time_connect>=T_pc % 中转成功，计算耗费时间（单位：分钟）
        sum_costTime_1=sum_costTime_1+T_pc*tickets.num;
        time1=T_pc;
      else
        num_transFail_1=num_transFail_1+1;
        sum_costTime_1=sum_costTime_1+360*tickets.num;
        time1=360; % 6小时
      end
      if time_connect>=T_pc+T_ts+T_wk
        sum_costTime_2=sum_costTime_2+(T_pc+T_ts+T_wk)*tickets.num;
        time2=T_pc+T_ts+T_wk;
      else
        num_transFail_2=num_transFail_2+1;
        sum_costTime_2=sum_costTime_2+360*tickets.num;
        time2=360;
      end
      if isLog
        LogTime =[LogTime; [tickets.num time1 time2]]; %
            记录顺序：乘客数目，问题2花费时间，问题3花费时间
        LogTense=[LogTense; [tickets.num time_connect time1/time_connect
            time2/time_connect]];
      end
    end
  end
  cost2=sum_costTime_1;
  cost3=sum_costTime_2;
  if isLog
    Log={LogTime LogTense num_scheduleSucess num_transFail_1
        num_transFail_2};
    disp('--------------分配结果----------------')
    disp(['总登机口数目：',num2str(length(GatesInfo))]);
    num_used=round(sum(sum(Y_ik,1)./(sum(Y_ik,1)+0.001))-1);
    disp(['使用固定登机口数目：',num2str(num_used)]);
    disp(['未被分配的飞机数目：',num2str(sum(Y_ik(:,K)))]);
    disp(['考虑的旅客订单总数：',num2str(length(Tickets))]);
    disp(['旅客订单成功分配数/占比(%)：',num2str(num_scheduleSucess), ...
    ' /',num2str(num_scheduleSucess/length(Tickets)*100),'%']);
    disp(['转乘失败旅客数目（问题2）：',num2str(num_transFail_1)]);
    disp(['转乘失败旅客数目（问题3）：',num2str(num_transFail_2)]);
    disp(['cost1:',num2str(cost1)]);
    disp(['cost2:',num2str(cost2),'分钟']);
    disp(['cost3:',num2str(cost3),'分钟']);
    else
    Log={};
  end
end
```

【*get_PassagerTime.m*】

功能：输入乘客订单中的到达和出发航班信息及分配的登机口，分别输出所需要花费的最小流程时间，捷运时间和行走时间。

```
%  输入：相对于乘客而言的离开航班所在转场记录，登机口信息GatesInfo
%       arrive_gate：欲达到的登机口
%  输出：旅客所用最小流程时间process 捷运时间transit 行走时间 walk，单位：分钟
if isempty(lea_gateIdx) || lea_gateIdx==70 || arr_gateIdx==70
  T_pc=0;T_ts=0;T_wk=0; return; %
      若飞机还未到达(为空)或被迫停在临时停机位中，则不予考虑
else
  arr_TorS=GatesInfo{arr_gateIdx}.TorS; % 到达的厅是 S型还是T型
  lea_TorS=GatesInfo{lea_gateIdx}.TorS;
  %       T_n  T_c T_s S_n S_c S_s S_e
  table_wk=[ 10 15 20 25  20  25  25 ;
         15  10  15  20  15  20  20 ;
          20  15  10  25  20  25  25 ;
         25  20  25  10  15  20  20 ;
         20  15  20  15  10  15  15 ;
         25  20  25  20  15  10  20 ;
         25  20  25  20  15  20  10]; % 单位：分钟
  if arr_planeType==-1 % 到达为国际航班
    if lea_planeType==-1 % 出发为国际航班
      table_pc=[20 30; 30 20];
      table_ts=[0 8 ; 8 0]; % 单位：分钟
    else % 出发为国内航班
      table_pc=[35 40; 40 45];
      table_ts=[ 0 8; 8 16]; % 单位：分钟
    end
  else % 到达为国内航班
    if lea_planeType==-1 % 出发为国际航班
      table_pc=[35 40; 40 35];
      table_ts=[0 8; 8 0]; % 单位：分钟
    else % 出发为国内航班
      table_pc=[15 20; 20 15];
      table_ts=[ 0 8; 8 0]; % 单位：分钟
    end
  end
  T_pc=table_pc(arr_TorS+1,lea_TorS+1); % 换算成以秒作为单位
  T_ts=table_ts(arr_TorS+1,lea_TorS+1);
  arr_wkidx=arr_TorS*3+GatesInfo{arr_gateIdx}.areaIdx;
  lea_wkidx=lea_TorS*3+GatesInfo{lea_gateIdx}.areaIdx;
  T_wk=table_wk(arr_wkidx,lea_wkidx);
end
```

```
end
```

【*gate_priority.m*】

功能：输入当前阶段下可分配的的登机口索引，根据设定好的决策规则，输出每个可分配登机口的权重，供局部决策使用。

```
% 输入：登机口信息，可用登机口序列
% 输出：根据重要性排列的登机口
% gateType: 0->DI, -1->I, 1->D
% 构造表格比较结果赋值表 ： I_DI_D
% 行表示 转场记录类型，列表示登机口类型
% ---------------------------
%    |  I  | DI  | D
% ---------------------------
%  I |  2  |  1  | 0
% ---------------------------
%  DI | 0  |  0  | 0
% ---------------------------
%  D |  0  |  1  | 2
% ---------------------------
  CalcTable=[2 1 0; 0,0,0; 0 1 2];
  k=length(gate_idxes);
  gate_weights=zeros(k,1);
  ftrans_type=[ftrans.arriveType ftrans.leaveType]+2;
  for idx=1:k
    gates_type_arrive=GatesInfo{gate_idxes(idx)}.arriveType+2;
    gates_type_leave=GatesInfo{gate_idxes(idx)}.leaveType+2;
    gate_weights(idx)=CalcTable(ftrans_type(1),gates_type_arrive)+ ...
    CalcTable(ftrans_type(2),gates_type_leave);
  end
  [~,ok_idx]=sort(gate_weights,'descend');
  ok_gates=gate_idxes(ok_idx);
end
```

【checkYik.m】

功能：检验决策变量是否符合要求，并返回统计结果

```
  [ FTrans,Tickets,GatesInfo]=data_read('filtered_data.xlsx');
  N=length(FTrans);
  K=length(GatesInfo)+1; % 包括临时机场
  % 已获得Y_ik后的计算检验
  disp('----------------------决策结果检验------------------');
  num_unassign=0; % 未分配飞机数
  Gates_available=zeros(length(GatesInfo),1); % 登机口可用时间
  for ft=1:N
    ftrans=FTrans{ft};
```

```
      choose_gate=find(Y_ik(ft,:)>0);
      if choose_gate>length(GatesInfo)
        disp(['ftrans:',ftrans.id,', 分配至临时停机场']);
        num_unassign=num_unassign+1;
      else
        gate=GatesInfo{choose_gate};
      if (ftrans.arriveType*gate.arriveType>=0 &&
          ftrans.leaveType*gate.leaveType>=0 ...
      &&
          ftrans.planeType==gate.planeType&&ftrans.arriveTime>=Gates_available(choose_gate))
      Gates_available(choose_gate)=ftrans.leaveTime+2700; % 更新时间
        disp(['ftrans:',ftrans.id,', 分配至:',gate.gateId]);
      else
        disp(['结果错误, 第',num2str(ft),'个转场记录']);
        result=1;
        return;
      end
    end
end

num_scheduleSucess=0; num_passagers=0; % 被安排到的旅客 % 旅客总数
num_transFail_1=0; num_transFail_2=0; % 未被安排到的旅客数
sum_costTime_1=0; sum_costTime_2=0; % 总体流程时间 %
    总体流程时间+捷运时间+行走时间
LogTime = [];    LogTense= [];  % 旅客换乘时间记录 % 旅客换乘紧张度记录
sum_Tense=0; % 问题3总体紧张度计算
for t=1:length(Tickets)
  ticket=Tickets{t};
  num_passagers=num_passagers+ticket.num;
  arrfIdx=ticket.FTrans_arriveIdx;
  leafIdx=ticket.FTrans_leaveIdx;
  arr_gate=find(Y_ik(arrfIdx,:)>0); % 乘搭的到达航班分配的登机口
  lea_gate=find(Y_ik(leafIdx,:)>0); % 乘搭的出发航班分配的登机口
  if arr_gate<70 && lea_gate<K
    num_scheduleSucess=num_scheduleSucess+ticket.num;
    [T_pc,T_ts,T_wk]=get_PassagerTime(
        GatesInfo,ticket.arriveType,arr_gate,ticket.leaveType,lea_gate);
    time_connect=ticket.connectTime/60;
  if time_connect>=T_pc % 中转成功, 计算耗费时间（单位：分钟）
    sum_costTime_1=sum_costTime_1+T_pc*ticket.num;
    time1=T_pc;
  else
    num_transFail_1=num_transFail_1+ticket.num;
    sum_costTime_1=sum_costTime_1+360*ticket.num;
    time1=360; % 6小时
  end
  if time_connect>=T_pc+T_ts+T_wk
```

```
        sum_costTime_2=sum_costTime_2+(T_pc+T_ts+T_wk)*ticket.num;
        time2=T_pc+T_ts+T_wk;
      else
        num_transFail_2=num_transFail_2+ticket.num;
        sum_costTime_2=sum_costTime_2+360*ticket.num;
        time2=360;
      end
      sum_Tense=sum_Tense+time2/time_connect;
      LogTime =[LogTime; [ticket.num time1 time2]]; %
          记录顺序：乘客数目，问题2花费时间，问题3花费时间
      LogTense=[LogTense; [ticket.num time_connect time1/time_connect
          time2/time_connect]];
      end
    end
    disp('-----------------无冲突发生，结果正确！--------------');
    disp([' 未分配飞机数 ： ',num2str(num_unassign)]);
    disp(['被使用登机口数： ',num2str(ceil(sum(sum(Y_ik,1)./(sum(Y_ik,1)+0.0001)))-1)]);
    disp(['考虑的旅客总数： ',num2str(num_passagers)]);
    disp(['安排到的旅客总数/占比(%)： ',num2str(num_scheduleSucess),'/',...
    num2str(num_scheduleSucess/num_passagers*100),'%']);
    disp('>>问题2');
    disp(['问题2中转失败旅客数： ',num2str(num_transFail_1)]);
    disp(['问题2总体流程时间（分钟）： ',num2str(sum_costTime_1)]);
    disp(['平均中转流程时间(分钟)： ',num2str(sum_costTime_1/num_scheduleSucess)])
    disp('>>问题3');
    disp(['问题3总体紧张度： ',num2str(sum_Tense)]);
    disp(['问题3中转失败旅客数： ',num2str(num_transFail_2)]);
    disp(['问题3总体流程+捷运+行走时间（分钟）： ',num2str(sum_costTime_2)]);
    disp('-------------------------------------------------');
    result=0;
end
```

【*test_Yik.m*】

功能：生成绘图所需中间变量

```
%%
% 1. 时间冲突检测
% 1.1 飞机安排时间约束
[ FTrans,Tickets,GatesInfo]=data_read('filtered_data.xlsx');
count = 1; K=70; N=303;
for k=1:K-1
  figure(2); hold on;
  for i=1:N
    if Y_ik(i,k)==1
      ftrans=FTrans{i};
      t=(ftrans.arriveTime-1516338000):10:(ftrans.leaveTime-1516338000);
```

```
        y=t*0+count;
        plot(t/60,y,'LineWidth',4.0); grid on;
      end
    end
    count = count+1;
end
ylim([0 y(1)+1]); xlim([0 3530]);
xlabel('时间(s)'); ylabel('登机口索引号')
%%
% 1.2.增加未分配飞机时间分布示意
count = 1;
for k=1:K-1
    figure(2); hold on;
    for i=1:N
      if Y_ik(i,k)==1
        ftrans=FTrans{i};
        t=(ftrans.arriveTime-1516338000):10:(ftrans.leaveTime-1516338000);
        y=t*0+count;
        plot(t/60,y,'LineWidth',4.0); grid on;
      end
    end
    count = count+1;
end
count=1;
for i=1:length(Y_ik(:,K))
    if Y_ik(i,K)==1
      ftrans=FTrans{i};
      t=(ftrans.arriveTime-1516338000):10:(ftrans.leaveTime-1516338000);
      y=t*0+69+count; count=count+1;
      plot(t/60,y,'LineWidth',4.0); grid on;
    end
end
ylim([0 y(1)+1]); xlim([0 3530]);
xlabel('时间(s)'); ylabel('登机索引号');

%%
% 2. 输出转场记录和登机口之间的对应关系
N=303; K=70;
GateVsFlight=cell(K,1); % 登机口对应的航班
FlightVsGate=zeros(N,1); % 航班对应的登机口
for i=1:N
    FlightVsGate(i)=find(Y_ik(i,:)>0);
end
for k=1:K
    GateVsFlight{k}=find(Y_ik(:,k)>0);
end
%%
```

```
% 3. 平均使用率计算
% 输出每个门的时间占比 Gate_time
Gate_time = zeros(1,69);
N = length(FTrans);
for i = 1:N
  ftran = FTrans{i};
  arrivetime = max(ftran.arriveTime,1516406400)/86400; % 转化为小时
  leavetime = min(ftran.leaveTime,1516492800)/86400;
  idx1 = find(Y_ik(i,:),1);
  if idx1 < 70
    Gate_time(idx1) = Gate_time(idx1) + leavetime - arrivetime;
  end
end

%%
[ result,LogTime,LogTense ] = checkYik( Y_ik );
% 3. 输出比率
% 3.1 中转旅客时空分布图
% 1) 问题2
N_dis1=ceil(max(LogTime(:,2))/5);
passDistri1=zeros(N_dis1,1); % 问题1旅客的时空分布
% 2）问题3
N_dis2=ceil(max(LogTime(:,3))/5);
passDistri2=zeros(N_dis2,1); % 问题2旅客的时空分布
num_passSum=0;
for i=1:length(LogTime)
  idx1=ceil(LogTime(i,2)/5);
  idx2=ceil(LogTime(i,3)/5);
  passDistri1(idx1)=passDistri1(idx1)+LogTime(i,1);
  passDistri2(idx2)=passDistri2(idx2)+LogTime(i,1);
  num_passSum=num_passSum+LogTime(i,1);
end
passDistriRation1=passDistri1/num_passSum;
passDistriRation2=passDistri2/num_passSum;
% 3.2 紧张度
% 1) 问题2
N_t1=ceil(max(LogTense(:,3))/0.1);
passTense1=zeros(N_t1,1); % 问题1旅客的时空分布
% 2）问题3
N_t2=ceil(max(LogTense(:,4))/0.1);
passTense2=zeros(N_t2,1); % 问题2旅客的时空分布
num_passSum=0;
for i=1:length(LogTense)
  idx1=ceil(LogTense(i,3)/0.1);
  idx2=ceil(LogTense(i,4)/0.1);
  passTense1(idx1)=passTense1(idx1)+LogTense(i,1);
  passTense2(idx2)=passTense2(idx2)+LogTense(i,1);
```

```
  num_passSum=num_passSum+LogTense(i,1);
end
passTenseRation1=passTense1/num_passSum;
passTenseRation2=passTense2/num_passSum;
```

5. 其他辅助函数

【planeType.m】-> 飞机宽窄 (0：窄机型，1：宽机型)，编码方式的好处：当拓展到考虑宽机型机位可停放窄机型机的时候，通过比较大小即可判断约束条件。

```
%  输入飞机类型的字符串，输出该飞机是窄机还是宽机
%  若为窄机则type=0，若为宽机则输出为type=1
  strWide={'332','333','33E','33H','33L','773','W'};
  strNarrow={'319','320','321','323','325','738','73A','73E','73H','73L','N'};
  planeStr=num2str(planeStr);
  for idx=1:length(strWide)
    if strcmp(planeStr,strWide{idx})
      type = 1; return;
    end
  end
  for idx=1:length(strNarrow)
    if strcmp(planeStr,strNarrow{idx})
      type = 0; return;
    end
  end
  disp('unfound:');disp(planeStr);
end
```

【flightType.m】-> 航班类型，国内或国外或国内外（对于登机口而言）-1：国际航班，0：国内国际航班（仅登机口可能等于该值），1：国内航班

编码方式的好处：当判断登机口航班类型与飞机航班类型是否相同的时候，只需要将两者的航班类型相乘，判断是否大于零即可，大于等于零则类型匹配。

```
%  输入航班类型的字符串，输出该航班是国内航班还是国际航班的编码
%  国内航班D则type=1，国际航班I则type=-1,国内国外航班均可DI，则type=0
flightStr=num2str(flightStr);
  str_cell={'I','D, I','D'};
  for idx=1:length(str_cell)
    if strcmp(flightStr,str_cell{idx})
      type = idx-2;
      return;
    end
  end
  disp('unfound:'); disp(flightStr)
end
```